摘 要

应用 CFD 研究汽车外流场，开发时间短，所用经费少，而且可以方便灵活的改变初始条件或者边界条件等，还可以获得流场中任意一点处的详细情况，因此应用 CFD 已经成为发展的必然。

流场的计算是一个反复迭代的过程，有计算量大，精度要求高，耗时较长的特点。而并行技术的引入为降低内存需要，减少计算时间，提高计算精度提供了可能的途径，因此，本文应用商业 CFD 软件实现了流场计算的并行化，并且对于并行计算的结果进行数据分析，计算出并行的加速比和计算效率。

首先，本文分别论述了汽车外流场的数值模拟以及并行计算的发展现状和必要性，介绍了 Fluent 软件，阐明了并行计算流体力学和商业软件的并行原理及其发展方向。

其次，本文介绍了使用商业 CFD 软件进行数值模拟以及并行计算的整个过程。先是对于汽车运动时外界的气流流动形成的流场，建立了模拟计算的数学模型，确定边界条件，并且进行了网格剖分，为了有大量的数据支持，本文对同一个问题划分了六组不同的网格数目，并对其进行计算比较。鉴于细分网格导致数值模拟计算量大，耗时较长，所以本文在课题组自己组建的 COW 上进行并行计算，分别对六组不同的网格数进行并行计算，得到大量的数据显示，并行计算大大缩短了计算时间。

Fluent 软件自带有很多种分区方法，本文选择两种分区方法进行比较。对并行计算结果评测有加速比、效率和可扩放性评测标准，本文详细介绍了这几种标准，并且针对加速比和效率方面对本文计算的问题进行了评测。

最后，提出了本文研究的不足和进一步研究的方向。

关键词： 汽车外流场；数值模拟；并行计算；加速比分析；

VISIT…

LANZAROTE Caliente.com

Abstract

When studying the flow field outside a car, the time is shortened and the cost is cut down using CFD. And it's also easy to change the start and boundary conditions, furthermore, the detailed result of any point in the filed can be got easily. So, it's necessary to use CFD in the filed of Fluid Dynamic.

The computing of the fluid flow is a process of iteration again and again, so that its' traits are big amount of computing, high requirement of precision and long time used. So, importing parallel technology is a good and efficient way to reduce the requirement of memory, shorten the computing time and improve the computing precision. Thus, we achieve the parallel fluid flow computing using commercial CFD software, analyze the data results and work out the speedup and the efficiency.

First of all, we discuss the development status quo and the need of simulation and parallel computing over the fluid flow outside the car, and, introduce the commercial software – Fluent, illustrate the parallel principle and the development direction of parallel CFD and commercial CFD software.

Next, we introduce the whole process of simulation and parallel computing using commercial CFD software. Model the fluid flow outside the car when the car is moving, define the boundary conditions and partition the mesh. In order to get large amount of data supporting the experiment, we compare 6 cases which have different meshes. Large meshes lead to large computing amount and long time, so we develop parallel computing base on the COW we organized. The data result displayed that the time used is shortened.

There are many methods of partition meshes, we choose two of them and compare them. There are three main methods evaluating parallel computing, they are speedup, efficiency and scalability, we introduce all these three methods in detailed, and we evaluate the cases on the side of speedup and efficiency.

Last, we advance the lack and the direction of the research.

Keywords: Fluid flow outside the car; Numerical simulation; parallel computing; speedup;

原 创 性 声 明

本人声明：所呈交的论文是本人在导师指导下进行的研究工作。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已发表或撰写过的研究成果。参与同一工作的其他同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签　名：辛晓华　日 期 2005.3.13

本论文使用授权说明

本人完全了解上海大学有关保留、使用学位论文的规定，即：学校有权保留论文及送交论文复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容。

（保密的论文在解密后应遵守此规定）

签　名：辛晓华　导师签名：　　　日期：2005.3.13.

第 1 章 绪 论

1.1 计算流体力学(CFD)

任何流体运动的规律都是由以下 3 个定律为基础的：质量守恒定律、动量守恒定律和能量守恒定律。这些基本定律可由数学方程组来描述，如，欧拉（Euler）方程和 N－S 方程。采用数值计算方法，通过计算机求解这些数学方程，研究流体运动特性，给出流体运动空间的定常或非定常流动规律，这样的学科就是计算流体力学。

计算流体力学（CFD，Computational Fluid Dynamics）是一门用数值计算方法求解流动方程以发现各种流动现象规律的学科，它的解是对微分方程精解的近似。它综合了计算数学、计算机科学、流体力学、科学可视化等多种学科。广义的 CFD 包括计算水动力学、计算空气动力学、计算燃烧学、计算传热学、计算化学反应流动、数值天气预报等。

自二十世纪六十年代以来 CFD 技术得到飞速发展，其原动力是不断增长的工业需求，而航空航天工业自始至终是最强大的推动力。传统飞行器设计方法试验昂贵、费时，所获信息有限，迫使人们需要用先进的计算机模拟手段指导设计，大量减少原型机试验，缩短研发周期，节约研究经费。四十年来，CFD 在湍流模型、网格技术、数值算法、可视化、并行计算等方面取得飞速发展，并给工业界带来了革命性的变化。如在汽车工业中，CFD 和其它计算机辅助工程（CAE）工具一起，使原来新车研发需要上百辆样车减少为目前的十几辆车；国外飞机厂商用 CFD 取代大量实物试验，如美国战斗机 YF-23 采用 CFD 进行气动设计后比前一代 YF-17 减少了 60%的风洞试验量。目前在航空、航天、汽车等工业领域，利用 CFD 进行的反复设计、分析、优化已成为标准的必经步骤和手段[1]-[2]。

1.2 数值模拟

所谓模拟（也称仿真），是指不直接研究现象或过程本身，而是用这些现象或过程相似的模型来进行研究的一种方法。模拟手段可分为物理模拟与数学模拟两类。

物理模拟是指在不同规模上再现某个现象，分析其物理特性和线性尺度的影响，可对所研究的过程进行直接实验。物理模拟多在按相似准则构成的实验室设备或中间实验设备上进行，即所谓相似模拟。

数学模拟是指用数学模型来使得现象或过程再现。因此，从广义上讲，表达

现象中的部分或全部的基本方程和表示自然规律的数学模型等都是数学模拟。从狭义上讲，数学模拟主要是指数字模拟，即把所研究的现象用数学模型表示出来，然后在数字计算机上对模型进行数值实验[3]。

1.3 汽车外流场的数值模拟

在传统的汽车空气动力学研究中，大多采用风洞实测的方法，但是其巨大的耗资对财力、规模要求甚高，在一般经济条件下根本无法进行。随着计算机和湍流理论的发展，从而把原来只能在风洞中进行的试验转化到计算机上来。与试验研究相比较，CFD 开发时间短，所用的经费少。应用 CFD 研究汽车外流场，在某种意义上比理论和试验对流体的运动过程认识的更为深刻，更为细致，不仅可以了解气流运动的结果，而且可以了解整体与局部的细致过程，能得到一些试验手段所得不到的理解和认识。CFD 可以方便灵活地改变初始条件、边界条件以及几何边界条件，并且可以获得整个流场中任意一点处的详细情况，使得清楚的研究流动机理变得方便可行。

1.3.1 国外状况

CFD 应用于汽车设计是从 80 年代以欧美为中心开始的。近年来，欧、美、日的汽车厂家已经开始致力于开发和利用 CFD 技术，取得了许多重要研究成果。

目前国外对汽车外流场的湍流解析模型主要是以 k-ε 模型、应力方程模型为代表的时间平均模型和空间平均的大涡模拟法（LES），计算所使用的网格数量已经达到 $2\times10^6\sim3\times10^6$，计算结果接近真实结果，误差控制在 5%以内。在结果分析中大量采用了可视化技术。由于计算所得的数据非常庞大、复杂，应用可视化技术可以凭借计算机把数字信息转化为易于研究人员分析和理解的图形或动画，加快对模拟对象的认识，提高工作效率。

目前，数值模拟的最主要问题就是计算精度问题。网格的形状、结构和所采用的湍流模型和计算方法都对精度有影响。进行整车的流场计算，通常需要划分几十甚至几百万个网格，这样的网格划分就是在大型计算机上计算也需要几天的时间来完成。因此，许多研究都集中研究网格生成技术。正在发展的多重网格法可以用更少的网格数、更少的计算时间以达到相同或者更高的计算精度。现在，国外空气动力学软件正朝着全自动划分网格、高速高精度计算的方面发展。

1.3.2 国内状况

由于我国的汽车工业发展一直比较落后，大多数厂家自行设计汽车的能力较

低，长期处于模拟或直接引进国外技术的开发状态，对汽车空气动力学的研究投入非常少。以往的空气动力学研究主要集中在飞行器和造船方面，汽车领域的应用研究很少涉及。

目前，在应用 CFD 进行汽车外形设计方面，国内已经有许多研究院把以前在航空、造船方面取得丰富经验利用起来，比较成功地对汽车外围流场进行了二维、三维的数值模拟，某些成果已经达到使用水平。例如，长沙交通学院与吉林工业大学合作，对轿车外围流场进行了二维数值模拟，采用 CFD 方法求解 N—S 方程，计算结果与风洞试验结果基本一致；南京航天航空大学用 Favre 平均 N—S 方程和 B/L 代数湍流模型对 Saab900 汽车外流场进行了数值模拟，分析了在不同车速下车尾上下气流分离的位置、尾涡涡心及尾涡长度的变化，并且得出了不同速度对流动阻力系数的影响，计算出汽车在 150km/h 下的流场矢量图和尾涡区的速度矢量图。

虽然国内一些研究成功地对汽车外流场进行了二维、三维的数值模拟，但是总的来说，国内这方面的研究还处于一个起步阶段。计算的对象还仅限于车体的基本形状，车身模型相对来讲比较粗糙，比如，车轮、保险杠、后视镜等都忽略了，而且划分的网格数目比较少，这些因素对计算结果与计算精度都会有一定的影响，研究的发展阶段相当于国外 80 年代初期的发展水平[5]-[6]。

1.4 并行计算(Parallel Computing)

计算流体力学应用领域要解决的问题越来越复杂，并且为了获得更为精确的结果，通常采用进一步细分网格，从而使得CFD分析的计算量变得更加庞大，计算能力要求也更高，而单CPU计算往往难以达到要求，所以迫切需要进行大规模并行处理的研究，从而并行计算能力也成为考核CFD软件的重要指标之一。

1.4.1 并行计算简介

并行计算（Parallel Computing）就是在并行计算机上所做的计算，它和高性能计算（High Performance Computing）、超级计算（Super Computing）是同义词，因为任何高性能计算和超级计算都离不开使用并行技术。

并行计算的发展是基于人们在两方面的认识：第一，单处理器的冯•诺伊曼机的性能已远远不能满足科学技术发展各方面的要求，这是因为它除了要受到顺序执行的限制，另外，VLsl 器件本身的开关速度也有一个极限，何况电信号的传播速度也受到光速的限制。因此，单单从提高 VLsl 器件本身的速度来提高冯诺伊曼机速度已经接近一个远低于人们要求的速度极限。因而，开发超级计算机已是社会发展、实现高性能计算、解决挑战性计算的需要。通过向量机或多处理

机实现并行处理是开发超级计算机的重要途径。第二，同时性和并行性是客观物质世界存在的普遍属性，具有实际物理背景的计算问题在许多情况下都可划分为能够并行计算的多个子任务。例如，根据几十个常规应用软件的统计，60%~80%的标量计算可以被向量化，而 90%左右的串行计算可以并行化[27]。

“计算”已经和“理论”与“实验”并列，被普遍认为是人类认识自然的三大支柱之一。这种计算，主要是指应用于科学与工程的、以高性能计算机为平台的大规模并行计算。它已经发展成为一门新的学科——大规模科学与工程计算。大规模并行计算已成为研究科学与工程技术问题的一种崭新的手段和方式，采用这种手段和方式进行的科学与工程技术问题研究，被称为“计算科学与工程”(Computational Science and Engineering)。著名的波音 777 飞机的设计，基本上就是依靠高性能计算机的“无纸设计”，它可以大量节省传统设计技术中昂贵的风洞实验，同时大大缩短设计时间[36]。

1.4.2 CFD 并行计算(Parallel CFD)

随着计算机技术的迅猛发展，CFD 得以迅速发展和普及。单机性能的提高使过去根本无法解算的问题在普通微机上可以解算，从而推动了 CFD 成为尖端工业、乃至一般过程工业的基本设计分析手段，从而大大激发了其应用。但是某些领域涉及大规模设计计算问题，不但单个作业计算量庞大，且需不断调整，重复计算。另一方面人们试图在很多场合期望放弃假设直接去求解和探索机理问题，如抛开基于统计理论的湍流简化模型直接模拟（DNS）湍流，这在计算能力有限的过去几乎是无法想象的。因此，CFD 研究人员发现硬件性能的提高随时会被无止境的需求所吞没，所以必须要做突破硬件性能的努力，尽管改进串行程序使用的数值算法可以提高计算效率，但并行化却几乎是唯一的大幅度提高计算效率的手段。

在 20 世纪 80 年代，向量计算机大大改善了 CFD 的计算速度，但是这种改善是以很高的费用为代价的。精简指令集计算机（RISC）在 20 世纪 90 年代的出现，提高了性能价格比，但是基于总线共享内存并行操作的规模不能超过 8 个处理器，原因是总线带宽限制了多 CPU 同时读取共享内存的速度。后来分布式内存（DMP）和分布共享内存（DSMP）机器的发展突破了这个瓶颈，Parallel CFD 得到迅速发展。Parallel CFD 一般采用物理区域分割并行方法，在编程上采用单控制流多数据流（SPMD）模型，采用 MPI 或 PVM 实现消息传递，几乎适用于所有的并行机体系结构：如向量机、MPP、集群系统、SMP 及其构成的星群系统，甚至是局域网连接的工作站/PC 机群[10]。

1. 国外 Parallel CFD 研究状况

在国际上软件研究先进的国家，计算流体力学领域并行算法研究开展较早。1993 年，德国的 Guy Lonsdal 和 Anton Sch Uller 用多重网格法对复杂流场模拟的三维 Navier-Stokes 方程进行并行计算，算法有较优的可扩展性，在 32 结点的 Intel iPSC/860 上，网格规模为 257×129 时的并行效率达到了 74%。1994 年，印度 Jawahala Nehru 先进科学研究中心的 Amit J.Basu 在有三个 i386 Floslver MK3 上使用谱方法对三维 Navier-Stokes 方程组织并行计算，在网格规模为 32^3 和 64^3 时的并行效率达到了 87.2%。1995 年，日本的 T.lwamiya 等在分布存储的数值风洞 NWT(Numerical Wind Tunnel)上实现了三维层流 Navier-Stokes 方程的并行计算，虽有比较多的全局通信，并行效率仍然比较高。同年，Cray 研究中心的 A.K.Stagg 等在 nCUBE/2、Intel iSPC/860、Intel Delta 等计算机上使用多种方法进行了三维 Navier-Stokes 方程的数值模拟并行计算，表明有较好的并行效率和可扩展性。1996 年，法国的 S.Lanteri 在 Intel paragon、Cray T3D 和 IBM SP2 等 MPP 机上实现了有限体积法和有限元法混合的四面体非结构网格程序 N3SMUSCL 求解喷气式飞机 FALCON 全机的并行计算，并行软件编程环境包括 PVM、NX 和 MPL，且比较了重叠网格与非重叠网格划分的并行效率，以及三种 MPP 巨型机的综合性能。

2. 国内 Parallel CFD 研究状况

在国内，北京航空航天大学的朱自强教授，在工作站机群环境下做了并行计算研究工作，得到了较高的加速比，并且扩大了模拟问题的规模。中国空气动力研究与发展中心和国防科技大学合作，在工作站机群上采用多重网格法并行求解跨声速 Euler 方程，12 台处理机上的加速比达到 9.06。与此同时中国空气动力研究与发展中心开展了三对角矩阵线性代数方程组化为无约束极值问题求解和七对角矩阵线性方程组并行算法研究工作。

近年来，中国空气动力研究与发展中心成立了 CFD 并行算法应用研究课题组，较广泛地开展了 CFD 并行算法技术研究。主要针对 CFD 特点，结合实际应用，基于分布式存储网络并行计算环境，开展并行算法研究，解决应用程序合理划分，降低冗余计算量，正确映射到并行处理系统，保证负载平衡，保证算法的可扩展性等显著并行效率的关键技术问题，研制 CFD 数值模拟并行软件，实现大规模高效并行计算，以解决工程型号设计中某些关键问题[12]。

1.5 课题的目的和意义

1.5.1 课题目的

随着汽车高速性能的不断提高，汽车外流场特性对汽车动力性、经济性、舒

适性和操纵稳定性的影响日益重要。通过分析汽车外流场特性对汽车造型的影响，可以减小汽车行进中所受的阻力，提高燃油经济性；降低空气噪声，提高乘坐舒适性；以及避免各种气动力对汽车操纵稳定性不良的影响等等。

用数值模拟的方法计算流体力学的问题，如果自己编程实现的话，从实现出计算软件再到使用此软件去计算，这个周期太长，所以本课题采用商业软件来计算。目前有国际上有很多非常优秀的商业数值模拟软件，从前处理、计算到后处理的整个过程都很成熟。

流场的计算是一个反复迭代的过程，有计算量大、精度要求高、耗时较长的特点。目前流场的计算主要还是限于数学模型、计算方法的改进，边界条件处理等方面，较少涉及到并行计算方面。而并行技术的引入为降低内存需要，减少计算时间，提高计算精度提供了可能的途径，因此，本文实现了流场计算的并行化，并且对于并行计算的结果进行了数据分析，通过加速比和计算效率的评测看出并行计算的优势。

1.5.2 课题意义

由于我们国家在基础科学上的研究底子薄、起点低，常常花大力气做了一些别人早已做过的工作。如今基础理论的研究可以与国外自由交流，别国的科研成果也可以为我们所用，而我们也该在已有的工作的基础上，合理的利用人力和资源，做进一步的研究。所以，本课题选择用成熟的商业 CFD 软件，而并非自己编程实现。选用 Fluent 的原因，是由于其软件设计针对每一种流动的物理问题的特点，采用适合于它的数值解法，在计算速度、稳定性和精度等各方面达到最佳。

目前，在国内外，对汽车外流场的计算主要还是限于数学模型、计算方法和网格剖分的改进，边界条件处理等方面，较少涉及到并行计算方面。本文将并行计算与汽车外流场的模拟结合起来，既能发挥数值模拟计算的优点，又有并行计算内存需求低、计算速度快、计算精度高的特点，考察了 Fluent 软件的并行计算能力，为进行大规模工程应用计算提供技术参照。

1.6 本文的研究内容

1.6.1 本文的内容安排

全文分为五章，其内容安排如下：

第一章：绪论

绪论部分综述了数值模拟和并行计算的内容，以及介绍了国内与国外对汽车

外流场的研究现状，最后提出本文的研究内容。

第二章：数值模拟及汽车外流场的前处理

本章节介绍了课题所使用的商用 CFD 软件——Fluent 软件，并且介绍了数值模拟的流程分为三部分：前处理、计算和后处理。这一章介绍的就是流程中的第一部分——前处理。对于在汽车运动时，外界的气流流动形成的流场建立了模拟计算的数学模型，确定边界条件，并且进行了网格剖分。

第三章：并行计算过程

对于传统的计算，流体力学计算量大、耗时长，所以采用并行计算来提高效率。本章首先介绍了并行计算的环境，并且对并行算法和 MPI 也作了简单介绍。主要介绍的是并行 CFD 的原理和并行 Fluent 的原理，以及在 Fluent 中的网格分区，也就是如何划分任务以分配给处理器进行并行计算。不同的分区方法，网格划分的方式不同，进行并行计算的时间开销也会不同，本文选用 Fluent 软件自带的两种分区方法：Principle Axes 方法和 Principle Strip 方法，在课题组组建的工作站 COW 上，分别对划分了六种不同网格数的汽车流场的问题进行计算，得到大量的数据。

第四章：后处理——可视化及数据分析

后处理对计算出的结果进行分析，用图形图像来显示是比较直观的。对并行计算得到的大量数据进行分析，评测并行计算的标准有加速比、效率和可扩放性评测，本文主要分析评测了加速比和效率两个方面，从而得出并行计算的优势。

第五章：总结

总结全文，并指出需要进一步研究的工作。

1.6.2 本文的主要工作

本文主要工作分为三个步骤，分别表现在第二、三、四章。第二章是前处理阶段，建立汽车外流场的数学模型并进行网格划分，设置边界条件；第三章是并行计算的阶段，对汽车外流场的问题进行并行计算；第三章是后处理阶段，对计算得到的大量数据进行分析，得出结果。

第 2 章　数值模拟及汽车外流场的前处理

2.1 数值模拟及商用 CFD 软件

2.1.1 数值模拟概要

现代流体力学研究方法包括理论分析、数值计算和试验研究三个方面。分析研究的优势是参数影响显示表述，可为数值计算和试验研究提供指导；而试验是认识客观现实的有效手段。数值计算则通过提供模拟真实流动的经济手段补充了理论及试验的空缺。数值模拟提供了廉价的模拟、设计、优化的工具，方便的提供全流场的详细信息，不需要过多的简化假设而能够给出所有参量的全部分布，对参数也没有什么限制，费用少，流场无干扰。

计算流体力学采用数值模拟方法来研究流体运动的基本物理特性。数值模拟（Numerical Simulations）是基于计算流体力学、计算传热学、计算燃烧学和冶金反应工程学的原理，应用数值计算方法，在计算机上求解联立的非线性质量、动量、能量及组分偏微分方程组，预报出流动、传热及夹杂物的运动过程的细节。

计算流体力学数值研究大体沿两个方向发展：一个是在简单的几何外形下，通过数值方法来发现一些基本的物理规律和现象，或者发展更好的计算方法；另一个则是为解决工程实际需要，直接通过数值模拟进行预测，为工程设计提供依据。

作为数值方法，计算流体力学不可避免地要采用一些近似处理方法：一是基于流动的物理现象对 N-S 方程的简化；另一类是对方程的离散及采用的解法技巧。相应地，其误差来源也就这三个方面，即：物理近似、数值离散近似以及代数方程迭代求解近似。

CFD应用数值模拟方法来研究流体运动的基本物理特性，其特点如下：(1) 给出流体运动区域内的离散解，而不是解析解，这区别于一般理论分析方法；(2) 它的发展与计算机技术的发展直接相关。这是因为可能模拟的流体运动的复杂程度、解决问题的广度和所能模拟的物理尺度以及给出解的精度，都与计算机速度、内存及输出图形的能力直接相关；(3) 若物理问题的数学提法（包括数学方程及其相应的边界条件）是正确的，则可在较广泛的流动参数（如马赫数、雷诺数、飞行高度、气体性质、模型尺度等）范围内研究流体力学问题，且能给出流场参数的定量结果。这常常是风洞实验和理论分析难以做到的[4]。

2.1.2 Fluent 软件简介

到目前为止，采用商用软件进行整个数值模拟的过程已经很完善，而优秀的CFD 商业软件也非常多，比如 Fluent，PowerFlow 和 Fine 等等，本课题组选用 Fluent 软件来进行计算。

在所有的 CFD 软件中，Fluent 软件是最著名的一个，在市场上占有 40%的份额。Fluent 包含多种燃烧模型、辐射模型和八种湍流模型，燃烧模型是 FLUENT 软件优于其它 CFD 软件的最主要的特征之一。而 Fluent 软件的前置处理器 GAMBIT 被认为是商用 CFD 软件最优秀的前置处理器，使用了非结构化网格技术和自适应技术，另外有丰富的 CAD/CAE 接口。Fluent 有强大的后置处理功能，能够完成 CFD 计算所要求的功能，包括速度矢量图、等值线图、等值面图、流动轨迹图、并具有积分功能，可以求得汽车所受到各种力、力矩及其对应的力和力矩系数、流量等。对于用户关心的参数和计算中的误差可以随时进行动态跟踪显示。现在，并行计算能力是考核 CFD 软件的重要指标之一。FLUENT 软件的并行功能具有以下三个特点：（1）自动分区技术：自动保证各 CPU 的负载平衡，在计算中自动根据 CPU 负荷重新分配计算任务；（2）并行效率高；（3）支持网络并行:支持网络分布式并行计算，内置了 MPI 并行机制。

Fluent 的软件设计基于 .“CFD 计算机软件群的概念”，针对每一种流动的物理问题的特点，采用适合于它的数值解法在计算速度、稳定性和精度等各方面达到最佳。不同领域的计算软件组合起来，成为 CFD 软件群，从而高效率地解决各个领域的复杂流动的计算问题，这些不同软件都可以计算流场、传热和化学反应，在各软件之间可以方便地进行数值交换，各种软件采用统一的前、后处理工具，这就为 Fluent 的通用化建立了基础。

由于囊括了 Fluent Dynamic International 比利时 Poly flow 和 Fluent Dynamic International（FDI）的全部技术力量（前者是公认的在粘弹性和聚合物流动模拟方面占领先地位的公司，而后者是基于有限元方法 CFD 软件方面领先的公司），因此 Fluent 软件能推出多种优化的物理模型，如，定常与非定常流动；层流（包括各种非牛顿流模型）；湍流（包括最先进的湍流模型）；不可压缩与可压缩流动；传热；化学反应等。对每一种物理问题的流动特点，有适合它的数值解法，用户可以对显示或隐式差分格式进行选择，以期在计算速度、稳定性和精度等方面达到最佳，Fluent 将不同领域的计算软件组合起来，成为 CFD 计算机软件群，软件之间可以方便的进行数值交换，并采用统一的前、后处理工具，这就省却了科研工作者在计算方法、编程、前后处理等方面投入重复、低效的劳动，而可以将主要精力和时间用于物理问题本身的探索上。在 Fluent5.0 中，采用 Gambit 的专用前处理软件，使网格可以有多种形式，对二维流动，可以生成三角形和矩形网

格；对于三维流动，则可生成四面体、六面体、三角柱和金字塔等网格；结合具体计算，还可生成混合网格，其自适应功能，能对网格进行细分或者粗化，或生成不连续网格、可变网格和滑动网格。总之，作为一个商品，Fluent 软件在一般工程技术中应用价值毋庸置疑[26]。

2.1.3 商用 CFD 软件的发展方向

CFD 在各种工业中的应用将仍然由全球几大 CFD 软件商主导。商业 CFD 软件主要发展方向是自动化功能，因为商业 CFD 软件的终极目标是提供给用户“好用”的分析工具以解决实际应用问题。自动化功能主要包括：

1. **网格的自动化生成**：每一个 CFD 应用至少三分之二的时间用于网格生成。网格生成障碍最根本的原因是计算能力的不足。设想若硬件能力无限大，任何一个问题都无需考虑针对具体几何结构的网格划分策略，只需按照足够精细的几何尺度进行均匀六面体网格生成，就能分辩实际工程问题的几何细节。因此商用 CFD 软件必须大力研发网格自动化生成功能，以减轻工程师的压力，缩短网格生成时间。

2. **网格自适应功能**：网格自适应的主要思想就是根据几何、流动变化等信息调整有限的网格用于流动模拟，使得采用较少的网格获得较高的流场精度。可见，自适应的压力最终也是来自于硬件能力的限制。网格自适应技术是解决硬件能力和求解精度矛盾的最好办法之一。但网格自适应必须基于基本准确的初始流场信息，否则会适得其反。

3. **输入参数敏感性的自动评估**：对于尖端工业如航空航天用户，计算的精度与可靠性是第一位的，尽管商业 CFD 软件经过大量的验证计算，但对特殊问题仍然必须检验其计算结果复现性。

4. **优化设计技术**：CFD 已完全具备求解用户设定的一个确定问题的能力，但是工业界对优化设计有越来越强的需求。目前“设计 → 分析 → 修改设计 → 分析”的过程仍然是不连续的、孤立的、人工干预的，而非自动的，每一次几何更改都带来大量的网格生成工作量，因此 CFD 软件商正在研发各种工业设备的定制模板，用户只要在模板中对主要优化变量设置约束条件，软件就能自动完成网格生成和计算。这称为“参数化”设计。在这儿值得提及“综合工程优化”（或集成工程优化，Integrated Engineering Optimization）的概念。众所周知，现实的

工程优化设计逼近过程中离散的中间态极多，而且一个最优设计不但是流动最合理的，还有结构最合理，甚至产品成本最低，设想人工重复每一个中间态的设计分析工作最终求得一个优化解的过程是多么漫长和昂贵。目前已经有这样的第三方优化设计软件，如 iSIGHT，扮演综合调度、优化算法的集成优化角色，它可以自动调度通用的流体分析、结构分析软件并由用户设置其它约束条件来完成一个特定优化问题的自动求解过程，使人工的干预降低到最低限度。

5.　基于大量计算实践生成和不断更新的专家库——知识引擎技术：每一个 CFD 应用工程师都知道，从一个生手变成熟手需要经过漫长的过程。除了专业基础外，同行间的交流、大量的实践经验的积累显得特别重要。CFD 应用的诀窍存在于每一个人脑中。目前著名 CFD 软件商美国 Fluent 公司正在构建一个专家库，把各个专家的实践经验和用户互动积累的经验提炼成一个专家系统，日后任何用户的应用问题都可以利用该专家系统寻求到最佳的问题解决模式。

6.　多场耦合：多场耦合一直是 CAE 领域极具挑战性的问题。在过去流体软件商、结构分析软件商各自为战，但自然界中很多的问题都是多场耦合问题。比如刚竣工的卢浦大桥在风场中风会对其产生激励引起振动，桥面的振动又影响了表面的风场结构，单独进行气动分析或强度分析都无法得到合理分析结果。而像航天飞机在飞行的时候，作用在机体的场有空气流场、气动加热温度场、过载、结构、等离子体产生的电磁场等。值得庆幸的是软件商们开始合作起来，首先在软件中互设数据接口，其次将不同场的分析模块整合成一套软件，或联合开发一些专用分析模块。多场耦合注定了计算量的庞大，并且并行化的难度随之增大。

2.1.4 使用 CFD 软件的数值模拟流程

使用商业 CFD 软件进行数值模拟，整个模拟过程分为三个部分：前处理模块、计算模块和后处理模块，如图 2-1 所示：

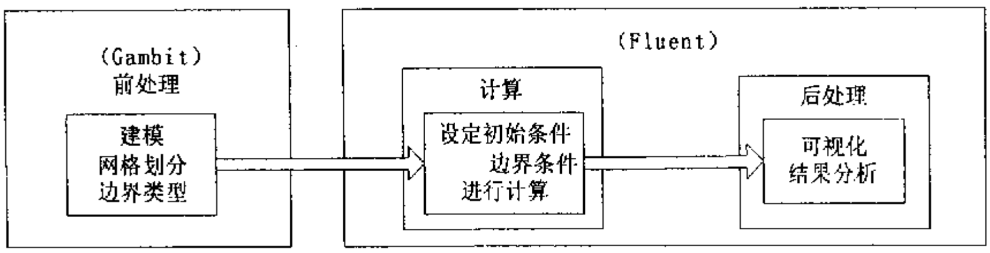


图 2-1　数值模拟流程图

1. 前处理

前处理模块的任务是在用户设置的各种参数的基础上，对研究对象进行建立适当的数学模型以及网格剖分，并对用户输入的物理参数进行处理，将其转换成一定格式的数据类型，提供给计算模块，交由 Fluent 来计算。前处理器使用 Gambit 来建模和划分网格，简单的模型直接用 Gambit 建立，复杂的模型则用 CAD 软件或 UG 软件来建立比较好，因为 CAD 建模能力比较强，建立模型后可以导入 Gambit 进行网格的划分。

2. 计算

计算则使用 Fluent 软件，Fluent 既可以在单机上使用，也可以在机群上做并行计算。它对前处理模块调用的控制方程进行离散化处理，并且加载初始条件和边界条件，然后通过求解器进行重复迭代计算，直到获得收敛的解为止，最后输出各节点的速度数据。

3. 后处理

后处理也就是可视化处理，仍然是由 Fluent 软件做，它负责处理计算模块输出的计算结果，将计算的结果以图形方式显示出来，从而比较直观和方便的分析计算的结果。

2.2 汽车外流场的前处理

2.2.1 流场控制方程

汽车外部流场一般是定常、等温和不可压缩的三维流场，数值模型一般为湍流模型[38]，理论上表达流体流动规律的方程是 N-S 方程，但目前由于计算机技术发展的限制，还不能完全求解 N-S 方程，以下采用工程中广泛应用的雷诺时均 N-S 方程：

$$(\overline{u}_i)_t + \overline{u}_j(\overline{u}_i)_{x_j} + (\overline{p})_x / \rho - \gamma\nabla^2(\overline{u}_i) + \overline{\rho}(\rho\overline{u}_i\overline{u}_j)_{x_j} = 0 \qquad (1)$$

$$(\overline{u}_j)_{x_j} = 0 \qquad (2)$$

其中，i，j=1，2，3，表示 3 个坐标轴方向；x_i，x_j 表示不同坐标方向的微分分量。

流场的主控方程为：

$$\frac{\partial \rho}{\partial t}+\sum_{i=1}^{3}\frac{\partial(\rho V_{x_i})}{\partial x_i}=0 \tag{3}$$

$$\frac{\partial \rho V_{x_i}}{\partial t}+\sum_{j=1}^{3}\frac{\partial(\rho V_{x_j}V_{x_i})}{\partial x_j}=\rho g_{x_i}-\frac{\partial P}{\partial x_i}+R_{x_i}+\sum_{j=1}^{3}\frac{\partial}{\partial x_j}(\mu_e\frac{\partial V_{x_i}}{\partial x_j})+T_{x_i}, \quad i=1,2,3. \tag{4}$$

其中，V_{x_i} 为速度分量，R_{x_i} 为分布流阻，ρ 为密度，g_{x_i} 为重力分量，T_{x_i} 为粘性损失项，μ_e 为有效粘度，$T_{x_i}=\sum_{j=1}^{3}\frac{\partial}{\partial x_j}(\mu\frac{\partial V_{x_i}}{\partial x_i})$，$\mu$ 为动力粘度[7]-[8]。

2.2.2 建立模型

进行汽车流场模拟，首要工作是车身模型的建立，如图 2-2 所示，由于车身是对称的，流场情况也是对称的，所以建模时只需建立一半的车身。车的原始模型首先是在 UG 上建立的，然后把车身几何模型文件的 cpt 格式转化成 igs 格式，再导入 Gambit 进行处理，做下一步工作。

使用 Gambit 在汽车模型周围建一个长方体区域作为计算域，为了使计算比较准确，计算域选取比较大，这样使得在流场仿真计算的时候，汽车流场就不受计算域的限制。

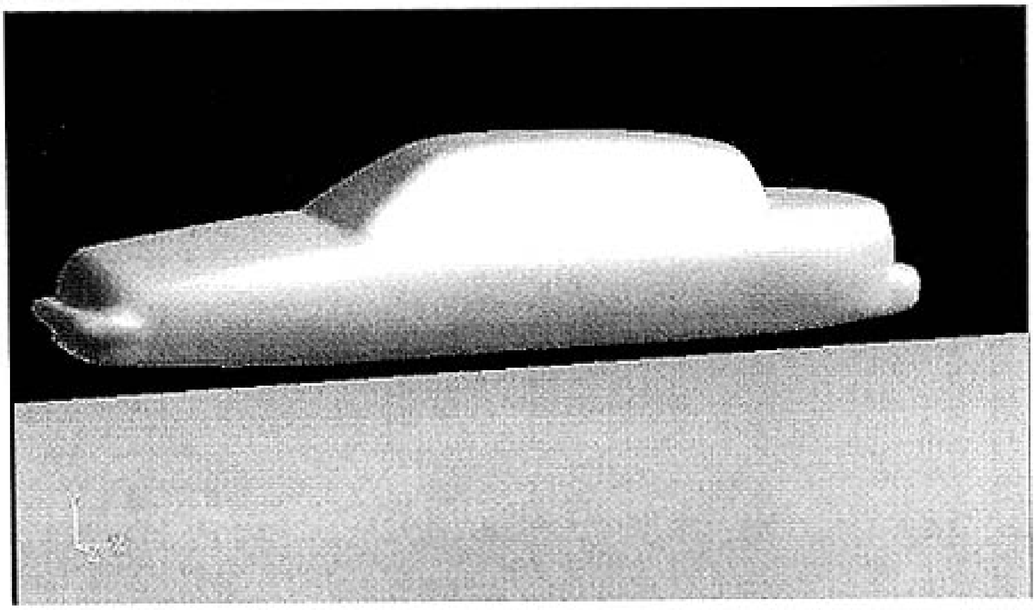


图 2- 2　汽车模型

2.2.3 网格结构

在选择网格的时候，需要考虑的主要有三个参数：初始化的时间、计算花费和数值耗散。

1．初始化时间

对于几何外形比较复杂的情况，如果采用结构网格或者块结构网格可能要花费很长的时间，甚至根本无法得到结构网格。由于复杂几何外形初始化时间的限制，所以在非结构化网格中使用三角形网格和四面体网格比较好。

2．计算花费

三角形网格和四面体网格所生成的单元会比等量的包含四边形网格和六面体网格的单元少得多。这是因为三角形网格和四面体网格允许单元聚集在流场的所选区域，而四边形网格和六面体网格会在不需要加密的地方产生单元。

3．数值耗散

多维条件下主要的误差来源就是数值耗散，又称为虚假耗散。之所以被称为虚假的，是因为耗散并不是真实现象，而是它和真实耗散系数影响流动的方式很类似。当流动和网格成一条直线时数值耗散最小，这个时候才要使用结构网格来计算。很明显，使用三角形/四面体网格流动永远不会和网格成一条直线，而如果几何外形不是很复杂时，四边形网格和六面体网格可能就会实现流动和网格成一条线。只有在简单的流动，如长管流动中，才可以使用四边形和六面体网格来减少数值耗散，而且在这种情况下使用四边形和流面体网格有很多优点，因为与三角形/四面体网格相比可以用更少的单元得到更好的解[9]。

由于汽车模型的几何外形比较复杂，而且要综合考虑到上面提到的三个因素，所以对于汽车模型网格区域的划分，先对车身进行面网格划分为非结构化三角形网格（如图 2-3 所示），接着对整个计算区域进行体网格划分为非结构化四面体网格（如图 2-4 所示）。

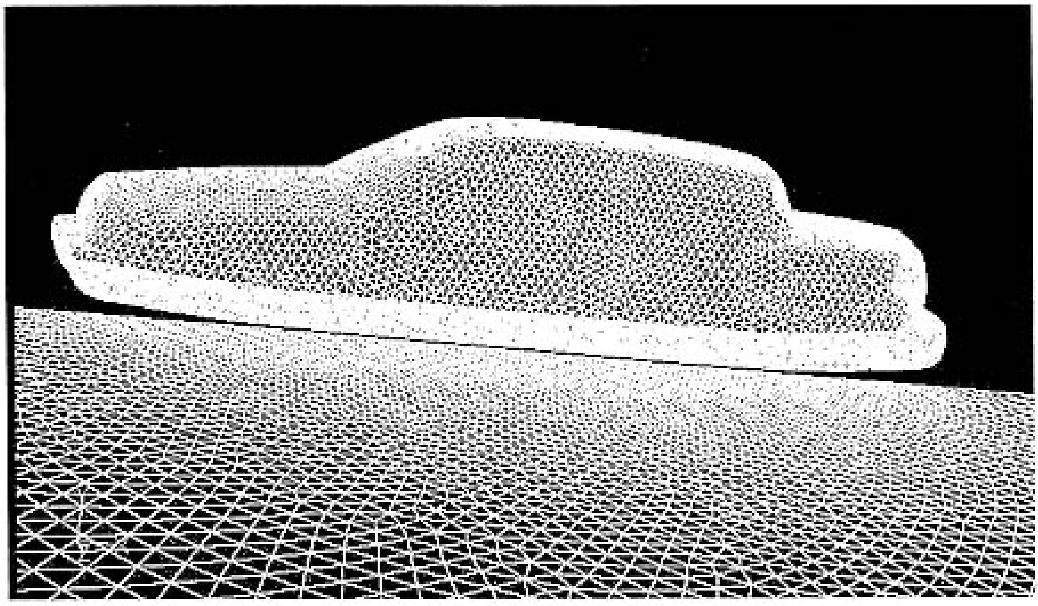


图 2-3　汽车表面网格

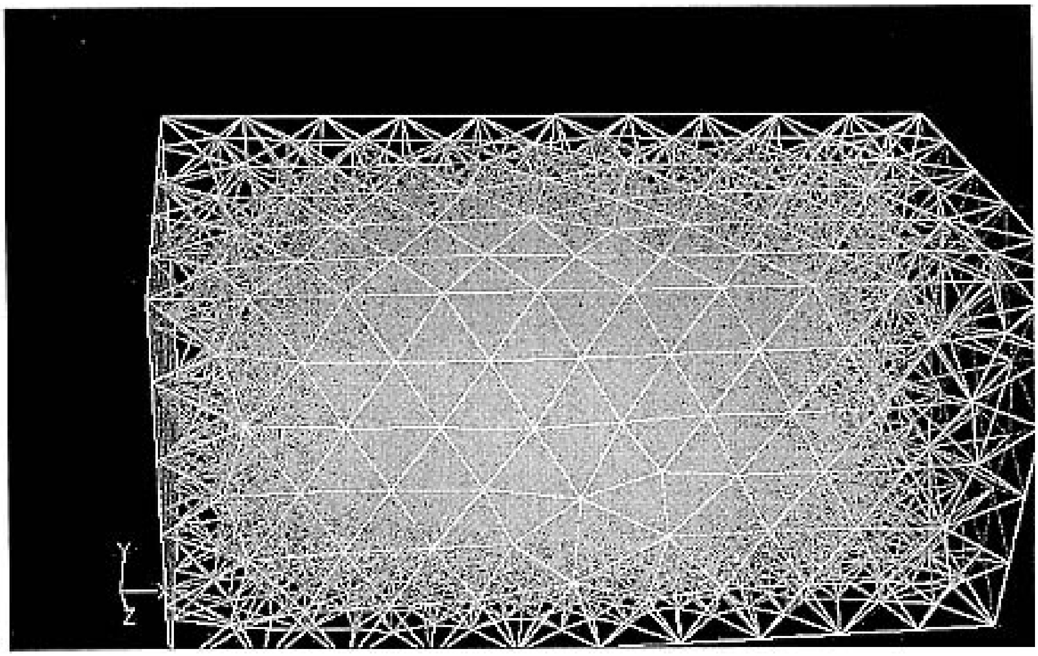


图 2-4　计算区域体网格

2.2.4 计算区域的边界条件

现实中的汽车是运动的，具有一定的速度，但是对于运动中的汽车进行模拟非常复杂，所以，本课题只对静止的汽车进行模拟，而让空气以一定的速度流向汽车，根据运动的相对性，处理实际效果是一样的。

流动边界定义为流体可进入或离开的界面，包括入口、出口及对称面：

1．入口

速度入口，设定空气流的水平速度为 20m/s，其他方向的速度梯度为零。压力在各个方向的梯度为零。

2．出口

压力出口，沿着出口方向的压力梯度为零。

3．对称面

在对称面上，除了垂直于边界的速度分量和雷诺应力变量之外，所用变量在数学上的对称的，边界没有扩散量，即平行于边界的物理量在对称面的法向上梯度为零。

4．车身表面(壁面)

速度在各个方向的梯度为零，压力的法向导数也为零[29]-[25]。

2.3 小结

本章第一小节介绍了数值模拟以及商用 Fluent 软件的一些基本知识，数值模拟分为三大部分：前处理、计算和后处理。第二小节开始做前处理，包括建立物理模型、划分网格、设置边界条件等。计算和后处理将会在后面的章节中介绍。

第3章 并行计算过程

3.1 并行计算简介

3.1.1 并行计算环境

并行计算就是以高性能计算机为平台的大规模计算。并行计算机主要指两台或两台以上的处理机连接起来可以并行操作的机器，亦可称之为并行多处理机。大型并行机系统一般可分为6类：单指令多数据流机SIMD（Single-Instruction Multiple-Data）、并行向量处理机PVP（Parallel Vector Processor）、对称多处理机（Symmetric Multiprocessor）、大规模并行处理机MPP（Massively Parallel Processor）、工作站机群COW（Cluster of Workstation）和分布共享存储DSM（Distributed Shared Memory）多处理机。SIMD计算机多为专用，其余5种均属于MIMD（Multiple-Instruction Multiple-Data）计算机。

当代主流的并行机是可扩放的并行机，包括SMP、MPP和COW：

共享存储的SMP系统结构具有如下特性：(1) 对称性：系统中任何处理器均可访问任何存储单元和I/O设备；(2) 单地址空间：OS可按工作负载情况在多个处理器上调度进程从而容易达到负载平衡，又因为所有数据均驻留在同一共享存储器中，所以用户不必担心数据的分配和再分配；(3) 高速缓存及其一致性：多级高速缓存可支持数据的局部性，而其一致性可由硬件来增强；(4) 处理器间的通信可用简单的读/写指令来完成。但SMP也有一些问题，比如欠可靠、可观的延迟、慢速增加的带宽、不可扩放性等。

MPP均使用物理上分布的存储器，且使用分布的I/O也渐渐变多。现今的MPP中，每个节点有一个或多个处理器和高速缓存（P/C）、一个局部存储器、有或没有磁盘和网络接口电路，他们均与本地互连网络连接，而节点间通过高速网络相连。早期的MPP主要应用于科学和工程超级计算，现在很多MPP都已成功地应用在商业和网络应用中。

工作站机群COW是实现并行计算的一种新主流技术，是属于分布式存储的MIMD并行计算机结构，由工作站和互连网络两部分组成，工作站上增加一块主机接口板用来实现连网。这种系统将一群工作站用某种结构的网络互连起来，充分利用各工作站的资源，统一调度、协调处理，以实现高效并行计算。互联网络可以是普通的LAN（如以太网、令牌环网和FDDI等），也可以是高速开关网络（如ATM、交换式高速以太网等）。工作站可以是高性能计算机，也可以是对称多处理机SMP。一个实用的COW还应有一个高效的软件环境，包括操作系统、通信

协议、可由用户调用的通信原语库以及并行程序设计环境与工具等。从用户角度看，COW相当于单一并行系统，感觉不到多个工作站的实际存在[11][35]。

3.1.2 并行计算

并行计算是一些可同时执行的进程的集合，这些进程互相作用和协调动作从而达到给定问题的求解。任何并行算法的设计都是基于某一特定的并行计算模型，而并行计算模型是从各种具体的并行机中抽象出来的。并行算法可分为数值计算的和非数值计算的并行算法，从其他的角度又可分为共享存储的和分布式存储的并行算法，或者同步的、异步的和分布式的并行算法。

求解基于代数关系运算的一类诸如矩阵运算、多项式求值、求解线性方程组等数值计算问题的算法称为数值算法（Numerical Algorithm）。非数值算法是指求解基于比较关系运算的诸如排序、选择、搜索、匹配等符合处理的非数值计算问题（Non-numerical Algorithm）[37]。

在分析并行算法时，通常要分析如下几个指标：

1. **运行时间 t(n)**

   算法运行在给定模型上求解问题所需的时间，通常包含计算时间和通信时间，分别使用计算时间步和选路时间步作为单位。

2. **处理器数 p(n)**

   它是求解给定问题所用的处理器数目。

3. **并行算法的成本 c(n)**

   并行算法的运行时间 t(n)与其所需的处理器数 p(n)之乘积，即 $c(n)=t(n)\times p(n)$。

4. **总运算量 W(n)**

   即并行算法所完成的总的操作数量。

3.1.3 MPI 简介

MPI(Message Passing Interface)[13]-[16]是目前发展较快、使用面较广的一个公共消息传递库。MPI 提供了一种与语言和平台无关，可以被广泛使用的编写消息传递程序的标准，它具有可移植性、高效性、灵活性和易用性的特点，不仅适用于具有分布式存储的大型机、工作站机群，也可用于具有共享存储的大型机。迄今为止，所有的并行计算机制造商都提供对 MPI 的支持，一个正确的 MPI 程序，可以不加修改地在所有的并行机上运行。

MPI 是一个库，而不是一门语言，对 MPI 的使用必须和特定的语言结合起来进行。而 MPI 很好的支持 FORTRAN 和 C 语言，在 MPI-2 中，除了和原来的

FORTRAN77 和 C 语言实现绑定之外，进一步与 FORTRAN90 和 C++结合起来，提供了四种不同的接口，为编程者提供了更多选择的余地。

3.2 并行计算流体力学(Parallel CFD)

3.2.1 并行原理

Parallel CFD 一般采用物理区域分割并行方法，在编程上采用单控制流多数据流（SPMD）模型，采用 MPI 或 PVM 实现消息传递，几乎适用于所有的并行机体系结构：如向量机、MPP、集群系统、SMP 及其构成的星群系统，甚至是局域网连接的工作站/PC 群。其并行原理是：将整个流动区域分割成 N 个子区域，分配给 N 个处理器计算，把子区域的初始流场信息、几何信息（网格坐标、标识号）分别载入各个子区域对应的 CPU 的内存中，在每一个 CPU 中启动计算进程，由主进程调度各 CPU 的计算。在每一次全场的扫描过程中，由各 CPU 完成子区域的计算并在边界完成数据交换（各 CPU 间的通信），由主进程收集全场数据完成收敛准则判别，并按照需要进行读写等其它操作。

在物理模型和数值算法确定的条件下，计算速度主要取决于 CPU 个数、CPU 性能、内存、CPU-内存访问带宽、结点互连带宽、网格质量及分区质量。每一个特定问题、每一台特定机器对应于一个最佳分区数，大量的实践会对同一类问题总结出一个最佳网格数/CPU 比值。分区数过多，CPU 间通信量增大，分区数增大到一定程度会反而降低计算速度；分区数过少，没有充分利用更多的 CPU 参与计算，也会影响计算速度；分区质量差，各 CPU 负载不均匀，CPU 有等待现象，也影响速度。通常成熟的商业软件包如 FLUENT 都至少提供两种分区方法：自动分区和手动分区。自动分区的好处在于操作简便，适宜工程应用，但各 CPU 的负载平衡不一定能保证。手动分区恰好相反。值得注意的是，对于稳态流动，在任何一个计算时刻内存中都只保留整场的数据信息，故在计算过程中只存在 CPU-内存、CPU-CPU（或计算结点-计算结点）之间的通讯，无需访问磁盘。但对于非定常问题，每一个时间步计算后均产生一套信息，所以每一时间步计算结束后必须将各计算结点的信息会聚起来，执行内存到磁盘的写操作，通信量骤增，计算速度无疑会放慢[28]-[34]。

3.2.2 发展方向

在 Parallel CFD 方面，发展高效率的并行计算方法是主要的努力方向。同时物理分区的并行计算属于非通信密集型，因此基于定制集群计算机和工作站/PC

联网计算将迅速得到推广，可以预测在未来几年中，国内将在航空、航天、汽车、船舶行业中完成该类并行计算的普及。

3.3 汽车外流场的并行计算

3.3.1 计算的软硬件环境

由于自强2000正在升级，所以本课题组自己组建工作站机群（COW），由五台DELL工作站通过以太网连接而成，共有12个CPU（Xeon 2G-2.4G），内存总量为7GB，每台工作站的硬盘容量为36GB-72GB，且每台工作站都有真实的IP地址。这个DELL工作站是专门做计算用，为了方便大家使用，采用远程登陆的方式，在客户端用RSH远程登陆。客户端的硬件配置为1个CPU，内存192MB，硬盘容量30GB。

之所以选择组建COW，是由于与SMP和MPP相比，COW在实用上具有一些明显的优点：（1）投资风险小：即使COW在技术上不够先进，但每台高性能工作站仍可使用，不会浪费资金；（2）编程方便：不需要学用新的并行程序语言，只要利用所提供的并行程序设计环境，在串行程序中相应的地方插入几条原语，便可使程序在COW上运行；（3）系统结构灵活：不同性能的工作站使用不同的体系结构和各种互连网络构成同构或异构的工作站机群系统，可更充分的满足各类应用要求；（4）性能/价格比高：一台高性能工作站相对便宜，而且一群工作站的总体运算性能接近一些巨型机的性能，而价格却低了很多；（5）能充分利用分散的计算资源：COW可在空闲时间内给空闲的工作站加载并行计算任务，从而充分利用资源；（6）可扩放性好：可根据需要增加工作站的数目，以高带宽和低延迟的网络技术支持获得高的加速比。

本课题组建的COW的软件环境则由Linux操作系统，采用RSH连接的并行编程环境MPI 1.2.5.2，商用计算软件采用Fluent6.1.22及其前置处理器Gambit2.1.6。客户端的软件环境为Linux Red Hat 9.0。

3.3.2 Parallel Fluent 的并行原理

Fluent的并行解算器是用多处理器来计算大型计算问题，计算可以在一台机器上执行，也可以在网络里的不同机器上执行。其原理与前面介绍的CFD的并行原理是一样的。划分区域的时候一般采用物理区域分割方法，如图3-1所示为要计算的流场区域，要把它划分成两部分，分别给计算节点0和1计算；图2为划分后的区域，由于两个计算节点间需要通信交换数据，所以在原来的块上面增加了

边界块。

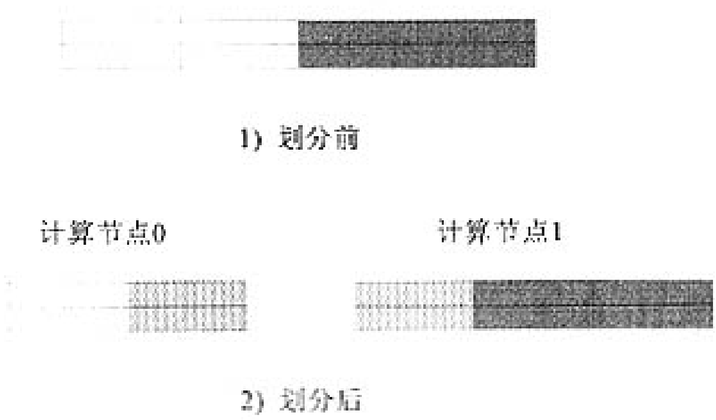


图 3-1 划分前后的计算区域

Cortex是主机Host的一个进程，Cortex将指令发送给计算节点0，计算节点0再把指令发送给其他的计算节点，并且与其他的计算节点是同步处理的。图3-2显示了并行Fluent中的进程间的关系。每个计算节点与其他的计算节点都虚拟联系着，计算节点间依赖于他们的“通信器”（communicator）来执行一些功能，比如发送、接收、同步和建立机器的连通性等等。Fluent的“通信器”实际上就是消息传递库一MPI（Message Passing Interface）标准。

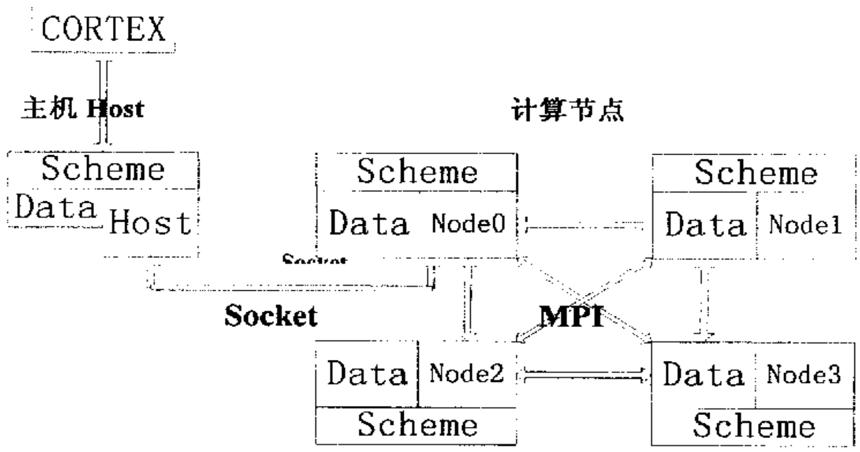


图 3-2 Fluent 的并行结构

3.3.3 FLUENT 网格分区

Fluent 并行计算就是利用多个计算节点(处理器)同时进行计算。并行计算可将网格分割成多个子域，子域的数量是计算节点的整数倍（如 8 个子域可对应于 1、2、4、8 个计算节点）。每个子域（或子域的集合）就会“居住”在不同的计算节点上。它有可能是并行机的计算节点，或是运行在多个 CPU 工作平台上的程序，或是运行在用网络连接的不同工作平台(UNIX 平台或是 Windows 平台)上

的程序。计算信息传输率的增加将导致并行计算效率的降低，因此在作并行计算时选择求解问题很重要。

3.3.3.1 Fluent 网格分区方法

在用 Fluent 的并行求解器时，需要将网格细分割为几组单元，以便在分离处理器上求解。网格分区有三个主要目标：

1. 生成等数量单元的网格分割；
2. 使分割的接触面数最小——减小分割边界面积；
3. 使分割的邻域数最小。

平衡分割(平衡单元数)可确保每个处理器有相同的负载，分割被同时传输。既然分割间的传输是强烈依赖于时间的，那使分割的接触面数最小就可以减少数据交换的时间。使分割的邻域数最小，可减少网络繁忙的机会，而且在那些初始信息传输比较长的信息传输更耗时间的机器来说尤为重要，特别是对依靠网络连接的工作站来说非常重要。

Fluent 里的分割格式是采用两分的原则来进行的，但并不像其他格式那样需要分割数，它对分割数没有限制，对每个处理器都可以产生相同分割数(也就是分割总数是处理器数量的倍数)。

网格采用两分法则进行分割。被选用的法则应用于父域，然后利用递归应用于子域。例如，将网格分割成四部分，求解器将整个区域（父域）对分为两个子域，然后对每个子域进行相同的分割，总共分割为四部分。若将网格分割成三部分，求解器首先将父域分成两部分——一个大概是另一个的两倍大——然后再将较大子域两分，这样总共就分为三部分。

并行计算机系统根据内存组织方式可以分为共享内存系统（Shared Memory System）和分布式内存系统（Network Memory System）。本课题组建的COW系统共有5台高性能计算机组成，命名分别为S1、S2、S3、S6和WZhang，S1机有4个处理器，其他的都是两个处理器。在设置Fluent并行的时候，如果用到一台机子并行的话，选择内存组织方式时为Shared Memory；如果用到两台以上计算机的话则选择为Network Memory方式。

由于本课题的网格数最多只有 30 万，所以最多仅使用了其中的 3 台来计算，对于两种方法分别计算以下五种情况：

1. 用 2 个 CPU 计算

使用 S1 节点机的其中两个 CPU 来运算；

2. 用 4 个 CPU 计算

使用 S1 节点机的全部 CPU 来运算，即四个 CPU 运算；

3. **用 5 个 CPU 计算**

   使用 S1 节点机的全部 CPU 和 S6 的其中一个 CPU 来运算；

4. **用 6 个 CPU 计算**

   使用 S1 节点机的全部 CPU 和 S6 的两个 CPU 来运算；

5. **用 8 个 CPU 计算**

   使用 S1 节点机的全部 CPU 和 S6 以及 S3 的两个 CPU 来运算；

3.3.3.2 两种分区方法的比较

Fluent 中有若干方法，有的方法只用于二维，有的则只用于三维。本实验采取了其中两种三维的分割方法进行比较，即 Principal Axes 方法和 Principal Strip 方法，这两种方法都属于两分法，分成两份的时候，两种方法结果一样，如图 3- 3 所示，用一个面将物体分为两部分。那么再往下分割便可看出其不同之处了：当进行一次分割时，Principal Axes 方法所用的分割面是垂直与前一次分割的面，而 Principal Strip 方法则是用平行于前一次分割面的面来进行分割。

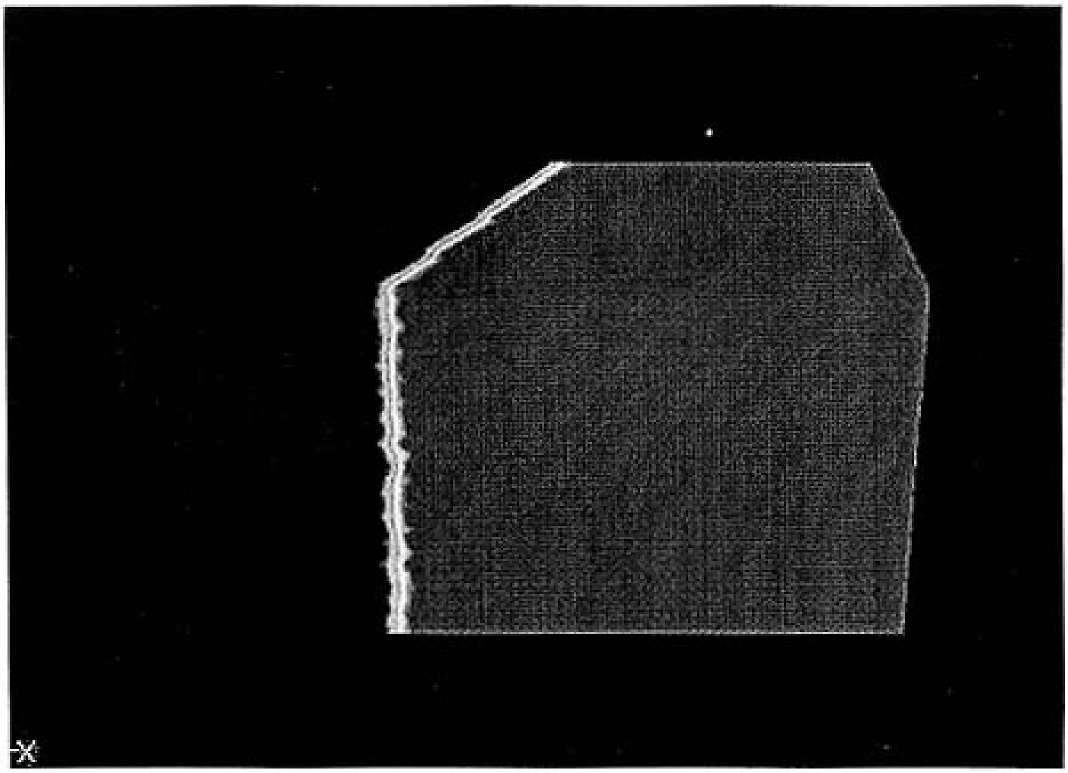


图 3- 3 用 Principal Axes 或 Principal Strip 方法产生的分割

如图 3- 4 所示，左边的一列是用 Principal Axes 方法产生的分割，右边的一列是用 Principal Strip 方法产生的分割。从上到下，依次分别分割为四份、五份、六份和八份。

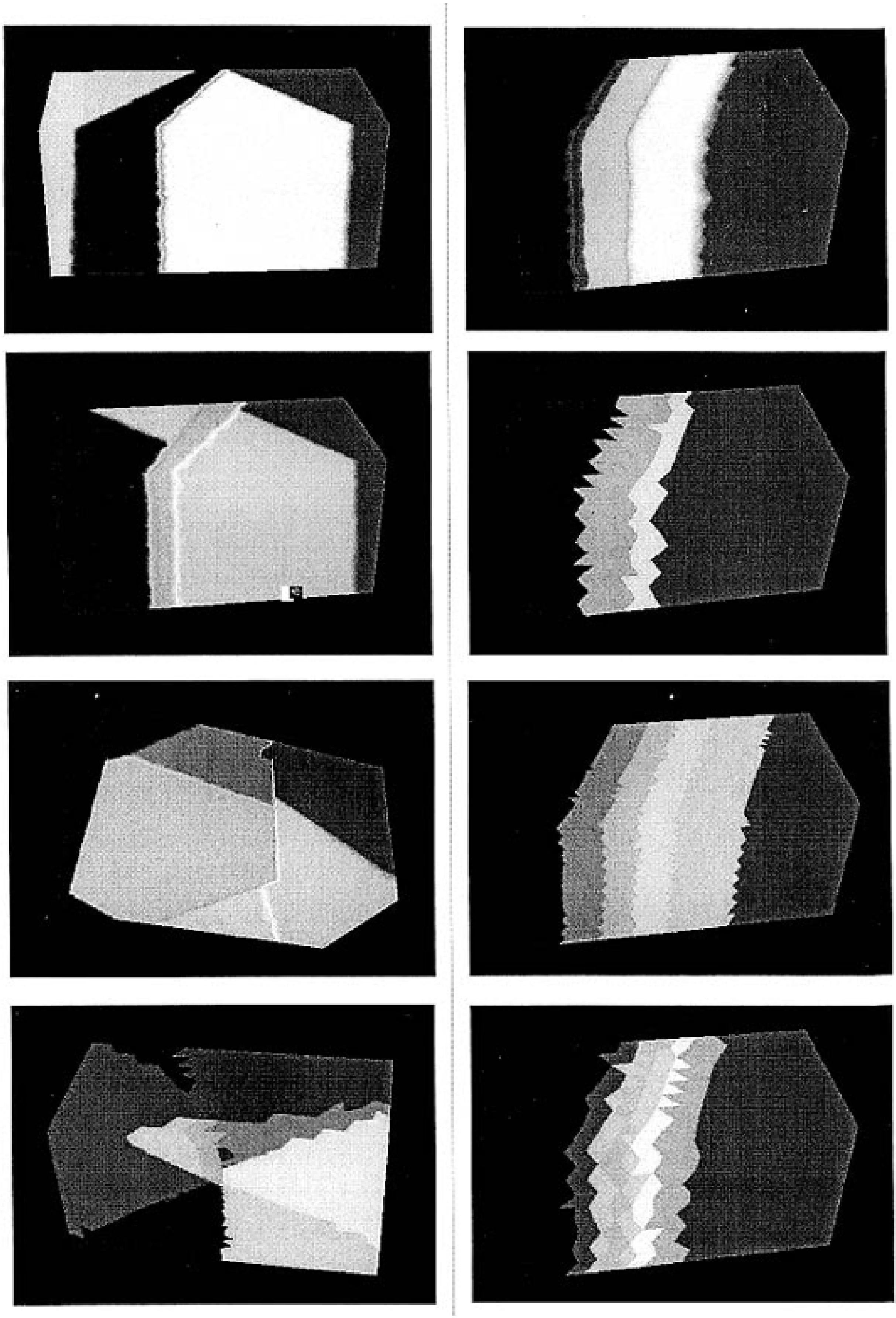


图 3-4　两种方法产生的分割比较

3.3.4 算例

本课题按划分的网格数不同分别进行了计算，网格被划分为 30 万、24 万、18 万、13 万、8 万以及 3 万的数量级。

3.4 Windows 环境下的并行计算

用课题组的 COW 是在 Linux 环境中计算的，只要 COW 组建好了，就可以用来计算。另外还实现了在 Windows 环境下，利用几台 PC 机来组建并行环境，实现 Fluent 并行，主要步骤如下：

1. 安装 Fluent

在一台计算机上安装 Fluent，并将这个 Fluent 的安装目录设为共享的；

2. 启动 RSH 服务

在每台计算机上都启动 RSH 服务：Fluent 自带有 RSH 的安装文件，拷贝到每台计算机的系统盘的 WINNT 目录下，在控制台中用命令“rsh –installed”进行安装，然后到控制面板的服务中启动 RSH 服务；

注意：通信机制完全可以不用 RSH，而改用 MPICH。

3. 加入域

在其中一台计算机上安装域控制器（必须注意的是 server 版的系统才能被设置为域控制器），然后其他的计算机都要加入到这个域中。安装域控制器的机器不一定必须是安装 Fluent 的那台机器；

4. 写 hosts 文件

将用作并行的所有机器名字都写到这个文件中，比如：

hosts.txt：

longar

lilymn

shu-ylf

写 hosts 文件时，有以下几点要注意：(1)第一台机子必须是安装 fluent 的机器；最后一行要留空。Hosts 文件并不是必需的，不过有这个文件的话，可以加快生成节点的速度。如果其中有的机器是多处理器的，那么这台机器名字要写两遍，当作两台机器一样写；(2)给这个文件取名的时候，不能有空格，比如：my host 就是错误的，而 myhost 是正确的。

5. 用控制台命令启动 fluent 并行版本

fluent *version* –t*num* –pnet [-cnf=*hostsname*] [-path*computername*\fluent.inc]
上面斜体的地方要换成自己需要的，*version* 是求解的问题是二维的还是三维的，或者是单精度还是双精度的；–t*num* 中的 *num* 指的是要用几台机器来做并行；–*cnf* 就是 4 中所写的文件；路径–*path* 是 hosts 文件所在的路径。
例如：

fluent 2ddp –t3 –pnet –cnf = hosts.txt –path\\longar\fluent.inc
2ddp 表示求解的问题是二维双精度的（二维单精度直接写成 2d 就可以了），-t3 表示要生成三个节点，从 hosts.txt 中位于前三位的机器选为做并行的，在启动了 Fluent 界面后还可以 kill 已生成的节点以及再添加节点。
中括号里的选项可以不写。

上面步骤中需要注意的是第 3 步，也是成功的关键。一般局域网的机器都会加入到一个 Group 中，用 Group 里的几台机器做 Fluent 并行时就会碰到问题了。原因是：Group 中的机器在互访时仍然需要访问的用户名和密码；而域中的机器只有在加入这个域的时候需要用户名和密码，当加入成功以后，机器间的互访就不需要进行验证了。

3.5 小结

本章一、二小节介绍了并行计算的一些知识以及并行 CFD 的原理和发展方向。第三小节介绍了数值模拟中的第二个部分一并行计算，包括课题的软硬件环境以及并行计算的网格分区。最后还介绍了 Windows 环境下做 Fluent 并行的步骤以及需要注意的地方。

第 4 章 后处理——可视化及数据分析

4.1 可视化结果分析

4.1.1 流线图

对计算的结果用 Fluent 自带的后处理功能以及 Tecplot 软件进行图形显示。将计算的结果文件（后缀为 cas 和 data 的文件）导入软件 Tecplot 进行处理，得到流线图，如图 4-1 所示，为汽车纵向对称平面上的气流流线图。从图中可以清楚的看到汽车周围的空气流的流动状态，气流在汽车顶面比较平滑，而尾部的涡旋又比较小，说明这个车型设计得比较好。

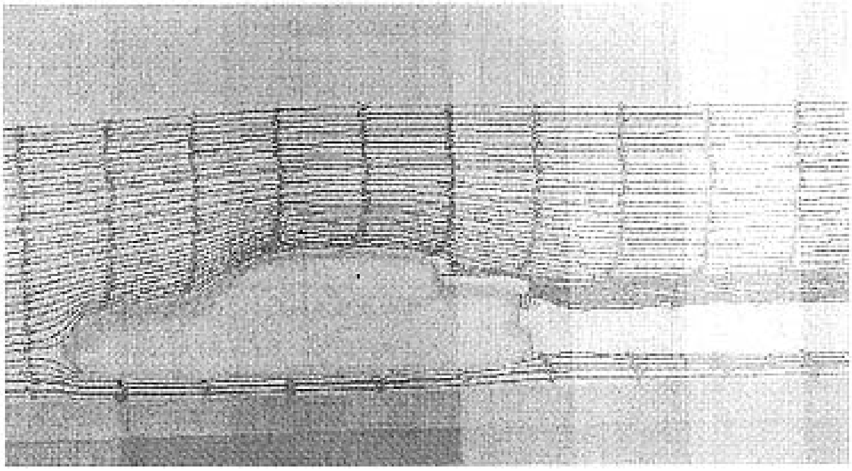


图 4-1 气流流线图

4.1.2 速度图

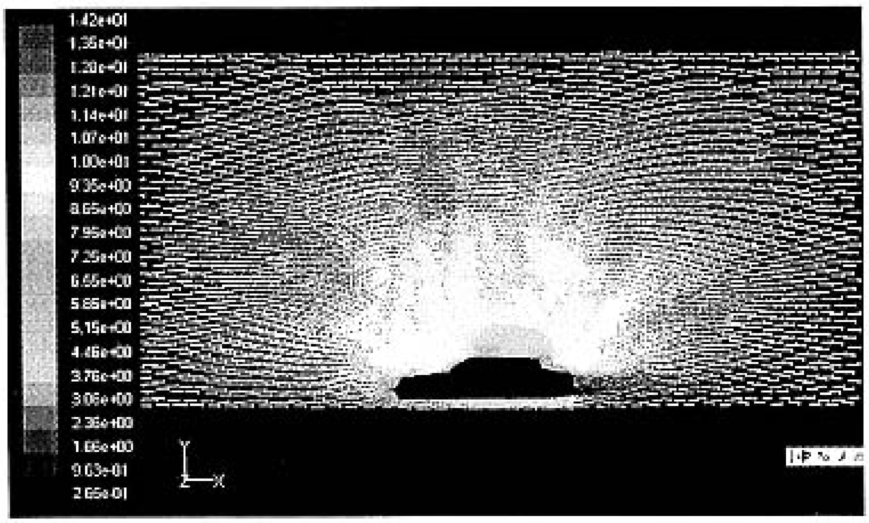


图 4-2 对称面速度云图

上面的图 4-2 是汽车对称面上空气流的速度表示图，从图中可以看出，汽车顶部的空气速度比较大，而汽车前端和后端速度比较小。下面图 4-3 和图 4-4 分别是汽车后窗和前窗的速度分布图，从图中可以看出，汽车后窗处和车尾两边的空气速度比较大，而车头和车尾中间的速度非常小。

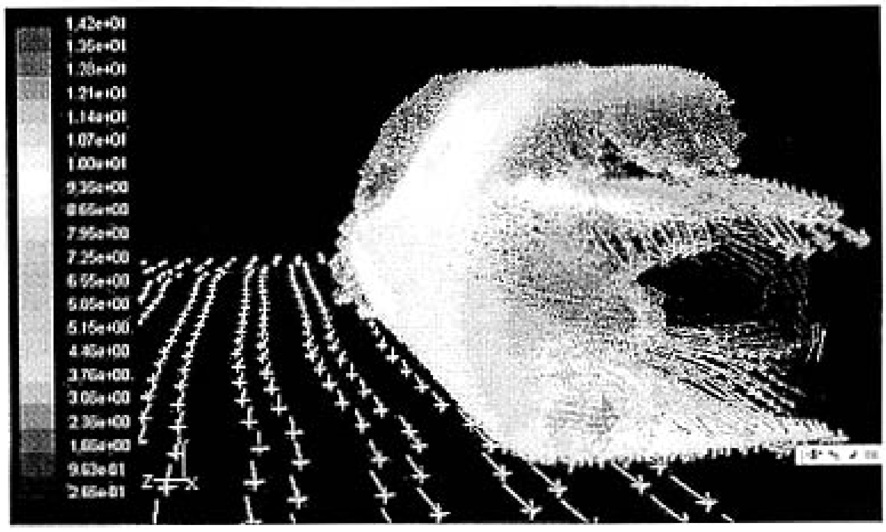


图 4- 3　后窗速度分布图

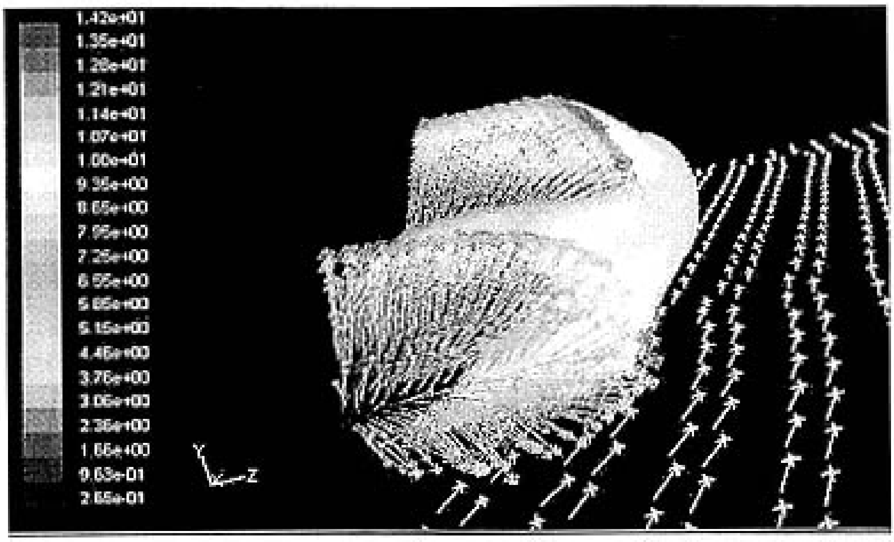


图 4- 4　前窗速度分布图

4.1.3 压力图

图 4-5 是汽车表面上的压力分布图，从图中可以看出，汽车最前端和前窗玻璃下部的压力比较大。

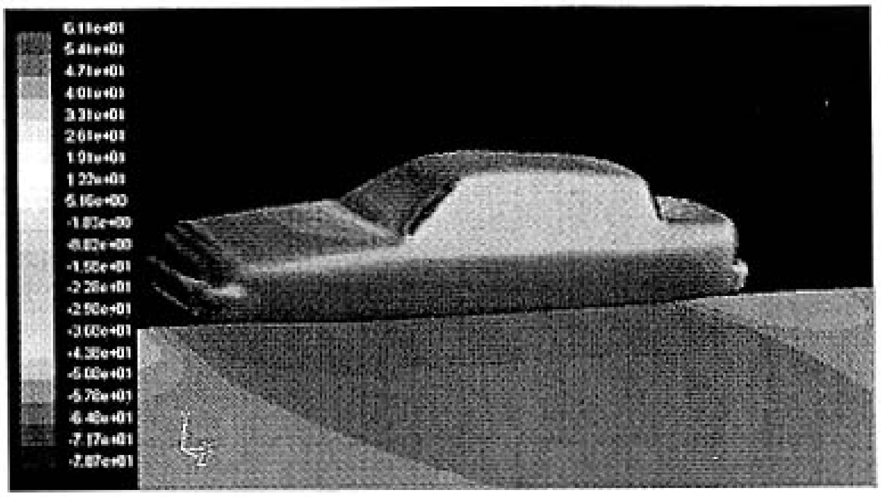


图 4-5　压力分布

4.2 并行计算性能评测

在并行系统上进行计算的主要目的就是要加速整个计算过程，所以研究并行系统的加速比性能是最根本的；随着计算负载的增加和机器规模的扩大，研究计算系统的性能是否能随处理器数目的增加而按比例的增加也是很重要的，这就是所谓的并行计算的可扩放性（Scalability）问题。

4.2.1 加速比及效率性能定律

并行系统的加速比是指对于一个给定的应用，并行算法的执行速度相对于串行算法的执行速度加快了多少倍。而根据科学计算的要求不同，有相应的三种加速比定律：（1）适用于固定计算负载的 Amdahl 定律；（2）适用于可扩放性问题的 Gustafson 定律；（3）受限于存储器的 Sun 和 Ni 定律。为了理解后面的可扩放性问题，要介绍一下这三种定律：

1.　Amdahl 定律

对于很多科学计算，实时性要求很高，所以时间是个关键因素，而计算负载是固定不变的。因此，在一定的计算负载下，为达到实时性，可利用增加处理器数来提高计算速度，而计算负载可分布到多个处理器上，这样增加了处理器就加快了执行速度，从而达到了加速的目的。

2.　Gustafson 定律

对于很多大型计算，精度要求很高，而计算时间是固定不变的。为了提高精度，必须加大计算量，相应地也必须增多处理器数目才能维持时间不变。

3.　Sun 和 Ni 定律

假若有足够的存储容量，并且规模可扩放的问题满足 Gustafson 定律规定的时间要求，那么就有可能进一步增大问题规模求得更好或更精确的解。

在实际应用中，并行实现的质量经常用得到的加速比和效率来衡量。一种并行算法在 p 个处理器上运行所得的并行加速比等于这个并行算法在单个处理器上运行时间与在 p 个处理器上运行时间之比。并行效率等于加速比除以 p 。

令 t_e^i 和 t_o^i 分别是并行系统上第 i 个处理器的有用计算时间和额外开销时间（包括通信、同步和空闲等待时间等）；所有 t_e^i 之和记为 $T_e=\sum_{i=0}^{p-1}t_e^i$ ，所有 t_o^i 之和记为 $T_o=\sum_{i=0}^{p-1}t_o^i$ ；令 T_p 是 p 个处理器系统上并行算法的运行时间，对于任意的 i ，显然有 $T_p=t_e^i+t_o^i$ ，且

$$T_e+T_o=pT_p \tag{1}$$

问题的规模 W 定义为由最佳串行算法所完成的计算量，也称为工作负载或工作量，即 $W=T_e$ ，则并行算法的加速比和效率分别定义如下：

$$S=\frac{T_e}{T_p}=\frac{T_e}{\frac{T_e+T_o}{p}}=\frac{p}{1+\frac{T_o}{T_e}}=\frac{p}{1+\frac{T_o}{W}} \tag{2}$$

$$E=\frac{S}{p}=\frac{1}{1+\frac{T_o}{T_e}}=\frac{1}{1+\frac{T_o}{W}} \tag{3}$$

从上面的式子看出：对于固定 p ， S 越大，说明并行算法处理时间越短；对于大多数的并行算法，由于算法中的并行度不高或缺乏负载平衡，以及通信、访问冲突和同步时间开销，使得效率 E 往往达不到 1，那么 E 越接近 1 则说明并行效果越好。

影响加速比的因素有：1)求解问题中的串行分量；2)并行处理所引起的额外开销（通信、等待、竞争、冗余操作和同步等）；3)加大的处理器数超过了算法中的并发程度。而增加问题的规模有利于提高加速比，原因是：1)较大的问题规模可提供较高的并发度；2)额外开销的增加可能慢于有效计算的增加（这句话最好换一种表达方式）；3)算法中的串行分量比例不是固定不变的（串行部分所占的比例随着问题规模的增大而缩小）。

4.2.2 可扩放性评测标准

可扩放性是指在确定的应用背景下，计算机系统（或算法或编程等）性能随处理器数的增加而按比例提高的能力。由上面介绍的三种加速定律可知，增加处理器数和求解问题的规模都可能提高加速比。一般情况下，增加处理器数，是会增大额外开销和降低处理器的利用率的，所以对于一个特定的并行系统、并行算法或并行程序，它们能否有效利用不断增加的处理器的能力是受限的，而度量这种能力就是可扩放性这一指标。

进行可扩放性研究的主要目的是：（1）确定解决某类问题何种并行算法与何种并行体系结构的组合，可以有效地利用大量的处理器；（2）对于运行于某种体系结构的并行机上的某种算法，根据算法在小规模处理机上的运行性能，预测该并行算法当移植到大规模处理机上后运行的性能；（3）对固定的问题规模，确定在某类并行机上最优的处理器数和可能获得的最大的加速比；（4）用于指导改进并行算法和并行机体系结构，以使并行算法尽可能地充分利用可扩充的大量处理器。

可扩放性度量有三种比较典型的方法：等效率、等速度和平均延迟方法。

1. 等效率度量标准

通常情况下，如果问题规模W保持不变，则随着处理器数p的增加，开销T_o也会随之增大，根据4.1，效率E也会相应下降。为了维持效率E不变，就要保持T_o/T_e的值不变，因此需要在处理器数p增大的同时相应地增加问题规模W的值，才有可能抵消由于p的增大而导致T_o增大的影响，从而保持效率不变。也就是说，为了维持一定的效率（介于0与1之间），当处理器p增大时，需要相应地增大问题规模W的值。由此定义函数$f_E(p)$为问题规模W随处理器数p变化的函数，称为等效率函数（ISO－Efficiency Function）。

按照等效率函数的定义，对于某一并行算法（或并行程序），为了维持运行效率保持不变，随着处理器数目的增加，若只需增加较小的工作量（即：问题规模），比如数W随p呈线性或亚线性增长，则表示该算法具有良好的可扩放性；若需增加非常大的问题规模，比如说W随p呈指数形式增长，则表示该算法是不可扩放的。

等效率度量标准可用简单的、可定量计算的、少量的参数就能计算出等效率

函数，并由其复杂性可指明算法的可扩放程度。这对于具有网络互连结构的并行机来说是很合适的，因为总的通信开销T_o是可一步一步计算出的。通信开销T_o通常包括了通信、同步、等待等非有效计算时间，但是在共享存储的并行计算机中，T_o则主要是非局部访问的读/写的时间、进程调度时间、存储竞争时间以及高速缓存一致性操作时间等，而这些时间都是难以准确计算的，所以用等效率函数度量可扩放性的方法就受到了限制。事实上，孙贤和与张小东于 1994 年分别提出了以试验测试为主要手段的两种评测标准，即等速度（ISO－Speed）和平均延迟（Average Latency）标准。

2. 等速度度量标准

在并行系统中，提高速度可以使用增加处理器数的方法，如果速度能以处理器数的增加而线性增加，则说明此系统具有很好的扩放性。令 p 表示处理器个数，W 表示要求解问题的工作量或称为问题规模，T 为并行执行时间，则定义并行计算的速度V 为工作量W 除以并行时间T：

$$V = W / T \tag{1}$$

而 p 个处理器的并行系统的平均速度定义为并行计算速度V 除以处理器个数 p：

$$\overline{V}=\frac{V}{p}=\frac{W}{pT} \tag{2}$$

根据上面的平均速度公式，定义等平均速度（简称为等速度）可扩放性度量标准如下：

对于运行在并行机上的某个算法，当处理器数目增加时，若增大一定的工作量能维持整个并行系统的平均速度不变，则成该计算是可扩放的。平均速度为常数，意即速度与处理器数目成线性比例增长。

按此定义，令W 是使用 p 个处理器的算法的工作量，令W' 表示当处理器数从 p 增大到 p' 时，为了保持整个系统的平均速度不变所需执行的工作量，则可得到处理器数从 p 到 p' 时平均速度可扩放度量标准公式为：

$$\psi(p,p') = \frac{W/p}{W'/p'} = \frac{p'W}{pW'} \tag{1}$$

用上述公式计算出的值介于 0 与 1 之间，值越大表示可扩放性越好。

当 $p=1$ 时，记 $T=T_1$；当处理器个数为 p' 时，记 $T'=Tp'$；则有：

$$\psi(1,p')=\frac{p'W}{W'}=\frac{T_1}{T_{p'}}=\frac{W}{W'/p'}=\frac{\text{解决工作量为}W\text{的问题所需串行时间}}{\text{解决工作量为}W\text{的问题所需并行时间}} \qquad (2)$$

上面的公式跟传统的加速比定义有点类似，其主要差别在于：加速比的定义是保持问题规模不变；而可扩放性定义是保持平均速度不变。也就是说，加速比是标志并行处理相对于串行处理所获得的性能增加；而可扩放性是标志从小规模系统到大规模系统所引起的性能衰减。

3. 平均延迟度量标准

参照图 4-6，令 T_i 为 P_i 的执行时间，它还包括在运行时所招致的延迟 L_i，程序运行时还包括启动与停止时间，T_o 是所有处理器的总的延迟时间。定义系统平均延迟时间 $\overline{L}(W,p)$ 为：

$$\overline{L}(W,p)=\sum_{i=1}^{p}(T_{para}-T_i+L_i)/p$$

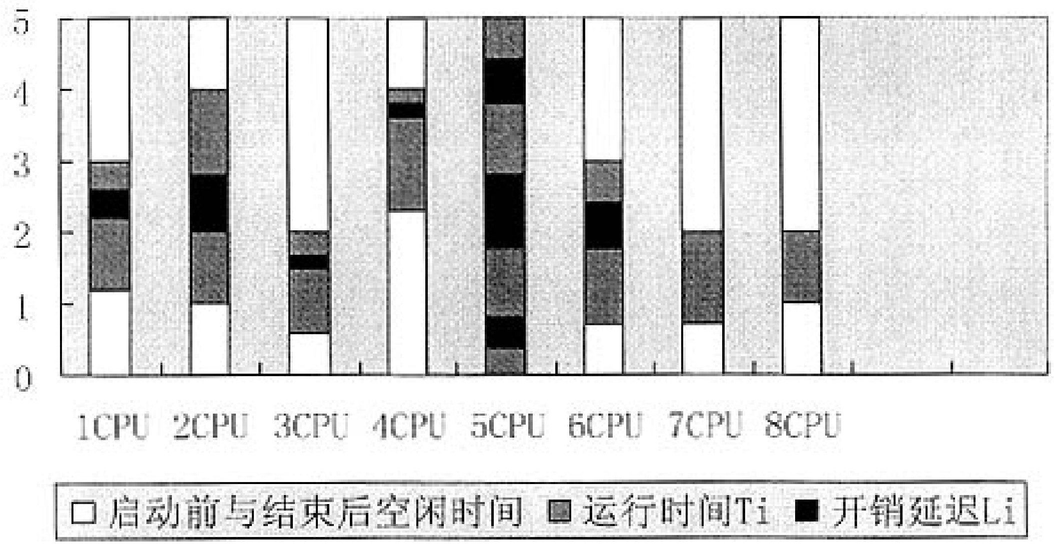


图 4-6　定义平均延迟示意图

因为 $pT_{para}=T_o+T_{seq}$ （T_{seq} 为串行执行时间）和 $T_o=p\overline{L}(W,p)$，所以

$$\overline{L}(W,p)=T_{para}-T_{seq}/p$$

对于一个算法—机器组合，令 $\overline{L}(W,p)$ 表示 p 个处理器上求解工作量为 W 问题的平均延迟；$\overline{L}(W,p)$ 表示在 p' 个处理器上求解工作量为 W' 问题的平均延迟。

当处理器由 p 变到 p'，而维持并行执行效率不变，则定义平均延迟可扩放性度量标准为：

$$\Phi(E,p,p')=\frac{\overline{L}(W,p)}{\overline{L}(W',p')}$$

用上述公式计算出的值介于 0 与 1 之间，值越大表示可扩放性越好[16]。

4.3 两种分区方法结果分析

4.3.1 Principal Axes 方法

4.3.1.1 数据结果

在并行计算中，从开始计算到计算结束所经历的时间称为总墙上时间；而每个 CPU 计算所用的时间和称为总的 CPU 时间。本实验并行计算每迭代步所执行的时间以及总墙上时间、总 CPU 时间皆如下所示：

（1）30 万网格的数据如表 4-1 所示：

表 4-1 网格为 30 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
1	S1	6.942	895.511	894.97
2	S1	3.956	510.333	1020.15
4	S1	2.107	280.202	1118.19
5	S1＋S6	1.744	275.57	1098.09
6	S1＋S6	1.449	194.149	866.48
8	S1＋S6＋S3	1.131	149.327	772.58

（2）24 万网格的数据如表 4-2 所示：

表 4-2 网格为 24 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
1	S1	5.093	893.052	895.93
2	S1	2.709	449.733	898.8
4	S1	1.521	249.369	994.62
5	S1＋S6	1.37	221.931	834.79
6	S1＋S6	1.027	171.497	786.08
8	S1＋S6＋S3	1.234	197.484	630.85

（3）18 万网格的数据如表 4-3 所示：

表 4-3　网格为 18 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
1	S1	3.712	486.546	490.563
2	S1	1.943	248.674	497.24
4	S1	1.101	140.941	560.82
5	S1＋S6	0.96	121.964	453.1
6	S1＋S6	0.755	95.193	418.63
8	S1＋S6＋S3	0.927	118.652	370.79

（4）13 万网格的数据如表 4-4 所示：

表 4-4　网格为 13 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
1	S1	2.391	496.846	501.125
2	S1	1.382	167.195	334.37
4	S1	0.746	90.257	357.47
5	S1＋S6	0.704	85.171	293.84
6	S1＋S6	0.571	69.149	273.704
8	S1＋S6＋S3	0.672	81.325	244.71

（5）8 万网格的数据如表 4-5 所示：

表 4-5　网格为 8 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
1	S1	1.385	301.26	302.43
2	S1	0.819	85.19	170.24
4	S1	0.448	40.752	161.6
5	S1＋S6	0.459	42.257	132.85
6	S1＋S6	0.375	34.463	124.92
8	S1＋S6＋S3	0.475	46.035	121.49

（6）3 万网格的数据如表 4-6 所示：

表 4-6　网格为 3 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
1	S1	0.408	41.62	42.39
2	S1	0.252	20.694	41.27
4	S1	0.143	11.87	47.11
5	S1＋S6	0.218	18.346	42.59
6	S1＋S6	0.19	17.317	44.27
8	S1＋S6＋S3	0.227	18.839	41

为了直观看出数据的结果，把计算执行时间的数据放到一起，如图 4-7 所示，可以看出总体规律是：(1) 随着网格数的增加，每步迭代执行的时间也是增加的；

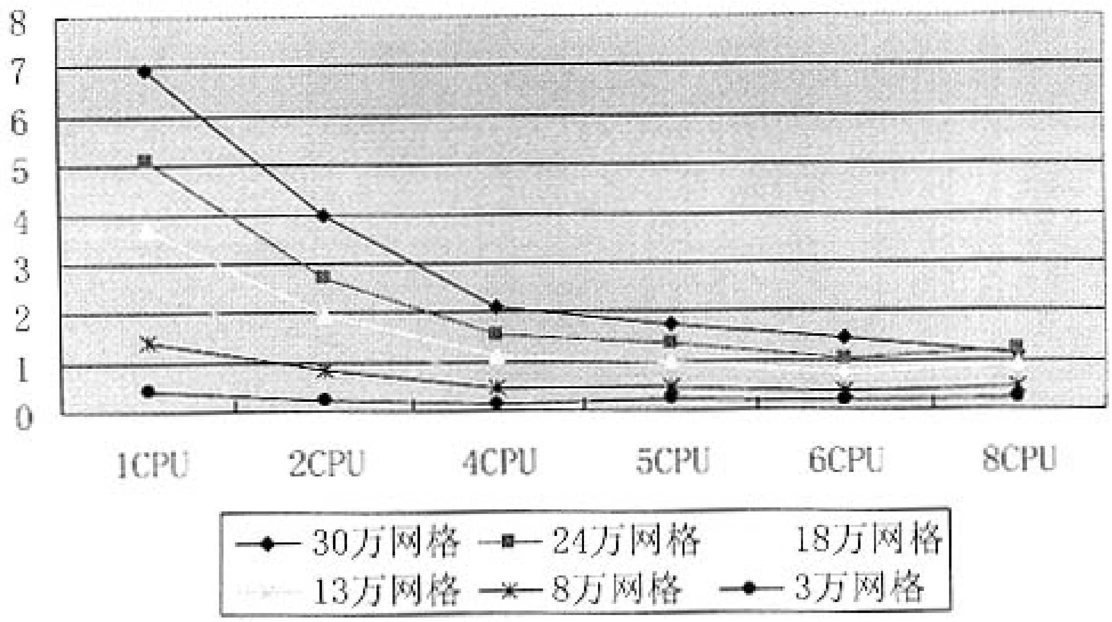


图 4-7　每步执行时间（s/步）

(2) 网格数越多，随着处理器的增加，每步执行时间缩短的比例也越明显；(3) 随

着处理器数的增加，每步迭代时间减小，但是并不是无限制的减小下去，当到了某个数目，之后反而会有所回升，比如：24 万网格数的例子，用 6 个 CPU 达到每步执行时间的最小值，用 8 个 CPU 的时候所用的时间比用 6 个 CPU 要多。

4.3.1.2 加速比分析

根据加速比的定义，加速比等于在多个处理器上执行的时间除以在一个处理器上的运行时间，如表 4-7 所示，计算出上面各种情况的加速比，例如：30 万网格数的问题，从表 4-1 中查出在一个 CPU 上的计算执行时间为 6.942s/步，在 5 个 CPU 上计算的执行时间为 1.744s/步，所以加速比为：6.942/1.744＝3.98，如表 4-7 中对应的数据。

表 4- 7 Axes 方法的加速比

结点数 / 网格数	2	4	5	6	8
30万网格	1.75	3.29	3.98	4.79	6.14
24万网格	1.88	3.35	3.72	4.96	4.13
18万网格	1.91	3.37	3.87	4.92	4
13万网格	1.73	3.21	3.4	4.19	3.56
8万网格	1.69	3.09	3.02	3.69	2.92
3万网格	1.62	2.85	1.87	2.15	1.8

为了更加方便直观的分析数据结果，用图示表示表 4-7 中数据，如图 4-8 所示为加速比分析图，从图中可以得出以下几点：(1) 网格数最大的情况，即 30 万网格数的情况，加速比随着处理器的增加而一直上升的；(2) 网格数为 24 万、18 万以及 13 万的情况，前面加速比也是随着处理器的增加而增大，在 6 个处理器处达到最大值，再后来又随着处理器的增加而有所减少；(3) 网格数为 8 万和 3 万的情况，在 5 个处理器时加速比下降，而 6 个处理器处又有所回升，8 个处理器处又开始下降。

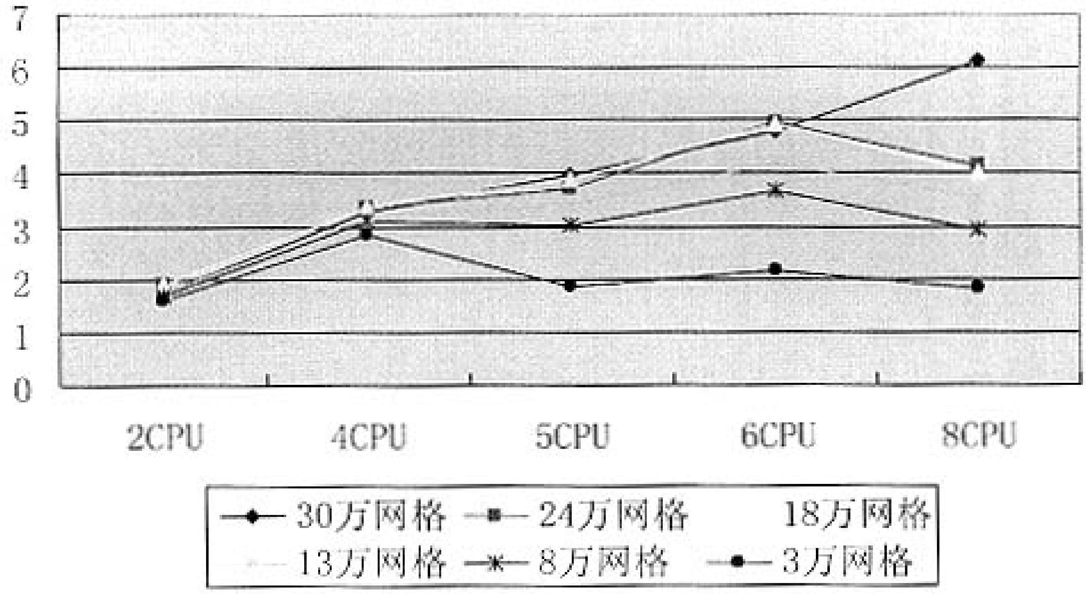


图 4-8 Axes 方法的加速比分析图

为了分析上面的图像所得到的现象，引入T_n的计算公式：

$$T_n = \sum_{1\le i\le k} \frac{T_1(i)}{\min(DOP_i, n)} + T_{par} + T_{interact}$$
$$= \sum_{1\le i\le k} T_n(i) + T_{par} + T_{interact}$$

其中，$T_n(i)$为第C_i步的并行执行时间，T_{par}和T_{interact}分别表示所有并行性和通信开销。由此可见，加速比的提高依赖于$\sum_{1\le i\le k} T_n(i)$和$T_{\text{interact}}$的减小。由于模拟计算的计算量比较大，数据间的相关性较小，因此，随着处理器的增加$\sum_{1\le i\le k} T_n(i)$呈线性减小，图中的各条曲线都呈现了较好的线性。而网格数为 30 万的情况，由于计算量的增大，并行性更好，曲线呈现了更好的线性关系。另外，随着结点数的增加，通信开销T_{interact}也随之增加，因此，加速比的增加逐渐放慢，尤其是计算规模较小的 3 万网格数的情况，表现比较明显而且较早的表现出来，当处理器达到 5 时，加速比就开始下降；而在处理器为 6 的地方又有所提高，原因在于用于计算的 COW，因为计算时用的前四个处理器是在节点机 S1 上，第 5 和 6 个处理器在节点机 S6 上，那么在一台节点机器内部的处理器间是共享存储的，那么就没有了通信开销T_{interact}，因此，这个地方的加速比有提高。

通过上面的分析，可以预测，当处理器继续增大到某个特定值时，网格数为 30 万的情况其加速比势必也会达到最大值，当处理器再继续增大，其加速比势必会降低。

4.3.1.3 效率分析

根据效率的定义，效率等于加速比除以节点数，可以理解为每个处理器的平均利用的程度。如表 4-8 所示，计算出上面各种情况的效率，例如，网格数为 18 万的情况，从表 4-7 中可以查到，用 5 个 CPU 计算的加速比为 3.87，那么其效率为：3.87/5×100%＝77.4%，如表 4-8 中对应的数据。

表 4-8　Axes 方法的效率

结点数 网格数	2	4	5	6	8
30万网格	87.5	82.25	79.6	79.83	76.75
24万网格	94	83.75	74.4	82.67	51.63
18万网格	95.5	84.25	77.4	82	50
13万网格	86.5	80.25	68	69.83	44.5
8万网格	84.5	77.25	60.4	61.5	36.5
3万网格	81	71.25	37.4	35.83	22.5

如下图的效率分析图所示，这里将单 CPU 的效率也列出来以方便比较，单 CPU 效率是 100%。可以看出随着处理器数的增加，效率的总体趋势是下降的，只是个别的地方会有浮动。网格数最大的情况其效率比较恒定；而其他的情况随着处理器的增加，效率极度降低，特别是 3 万网格的情况，用 8 个处理器时效率竟然才达到 22.5%，所以，如果网格数比较少，用并行计算非常浪费处理器资源。

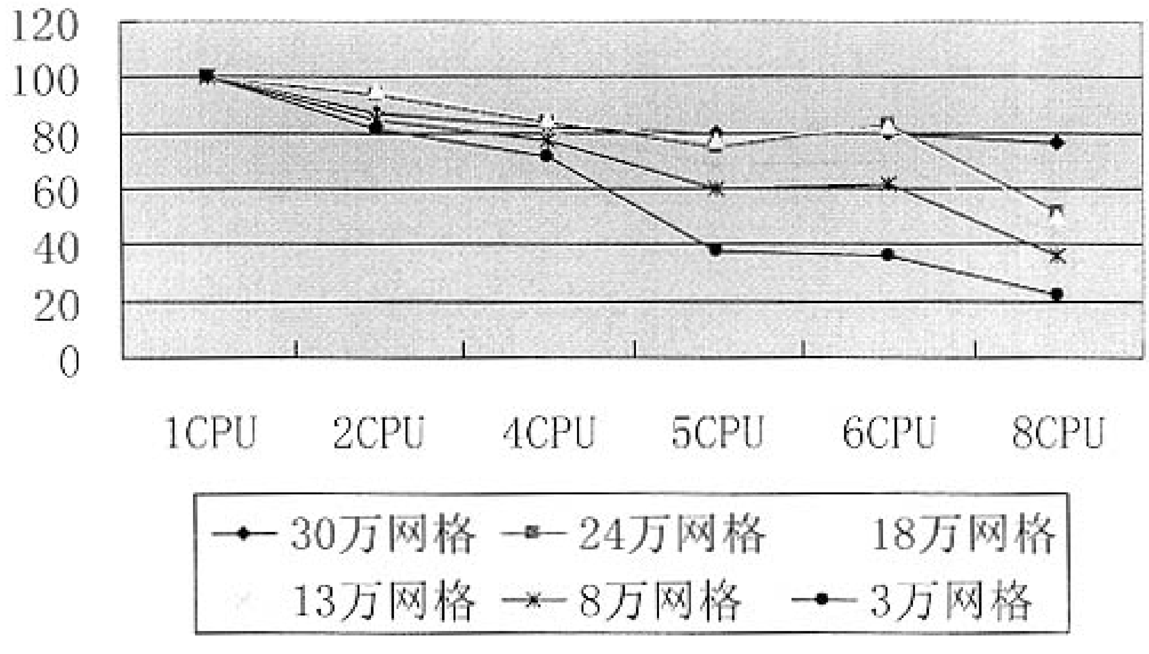


图 4-9　Axes 方法的效率分析图

4.3.2 Principal Strip 方法

4.3.2.1 数据结果

（1）30 万网格的数据如表 4-9 所示：

表 4-9 网格为 30 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
2	S1	4.153	576.542	1153.762
4	S1	2.231	291.103	1164.513
5	S1＋S6	1.755	226.411	909.58
6	S1＋S6	1.575	203.149	796.64
8	S1＋S6＋S3	1.667	215.01	676.25

（2）24 万网格的数据如表 4-10 所示：

表 4-10 网格为 24 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
2	S1	2.613	413.726	827.203
4	S1	1.423	215.163	860.701
5	S1＋S6	1.251	208.846	847.9
6	S1＋S6	1.221	199.062	749.81
8	S1＋S6＋S3	1.252	209.024	645

（3）18 万网格的数据如表 4-11 所示：

表 4-11 网格为 18 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
2	S1	1.913	237.623	475.16
4	S1	1.09	140.585	561.05
5	S1＋S6	0.865	109.896	448.76
6	S1＋S6	0.862	109.533	405.47
8	S1＋S6＋S3	0.939	120.142	358.24

（4）13 万网格的数据如表 4-12 所示：

表 4- 12　网格为 13 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
2	S1	1.293	160.186	320.612
4	S1	0.735	88.913	561.05
5	S1＋S6	0.653	80.12	400.62
6	S1＋S6	0.608	74.23	267.47
8	S1＋S6＋S3	0.687	83.171	242.21

（5）8 万网格的数据如表 4-13 所示：

表 4- 13　网格为 8 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
2	S1	0.855	88.924	177.68
4	S1	0.422	38.443	153.47
5	S1＋S6	0.415	37.698	133.25
6	S1＋S6	0.401	36.865	122.16
8	S1＋S6＋S3	0.471	43.303	115.79

（6）3 万网格的数据如表 4-14 所示：

表 4- 14　网格为 3 万的计算时间

计算方案（节点数）	计算节点（机器名）	计算执行时间（s/步）	总墙上时间	总CPU时间
2	S1	0.269	22.034	44.06
4	S1	0.138	12.681	50.47
5	S1＋S6	0.17	14.112	40.58
6	S1＋S6	0.185	15.536	39.86
8	S1＋S6＋S3	0.23	19.074	40.88

为了便于比较，把计算执行时间的数据放到一起，如图 4-10 所示，将曲线结果与 Principal Axes 的结果比较，即与图 4-7 进行比较，发现结果和 Principal Axes 方法得出的结果很类似，曲线的走势也差不多。

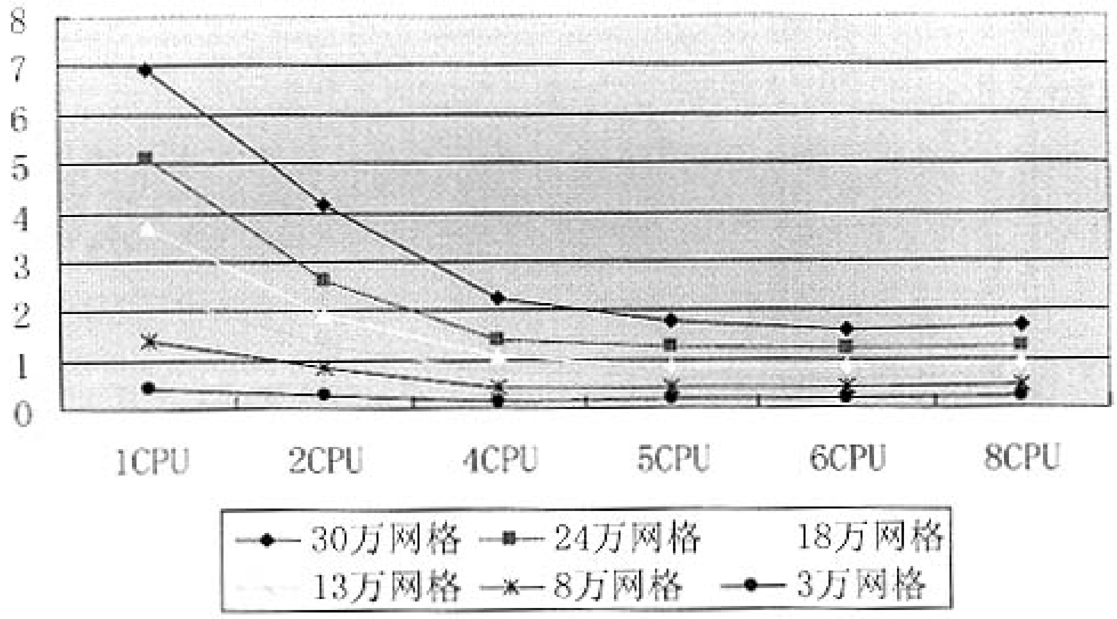


图 4-10 每步执行时间（s/步）

4.3.2.2 加速比分析

表 4-15 Strip 方法的加速比

结点数 网格数	2	4	5	6	8
30万网格	1.67	3.11	3.96	4.41	4.16
24万网格	1.95	3.58	4.07	4.17	4.07
18万网格	1.94	3.41	4.29	4.31	3.95
13万网格	1.85	3.25	3.66	3.93	3.48
8万网格	1.62	3.28	3.34	3.45	2.94
3万网格	1.52	2.96	2.4	2.21	1.77

与 Axes 方法一样，计算出加速比，如表 4-15 所示，转换成图像如图 4-11 所示，对比两种方法得到的加速比图可以发现，整个曲线的走势是类似的，具体每条曲线走势严格不一样，这是合乎情理的。

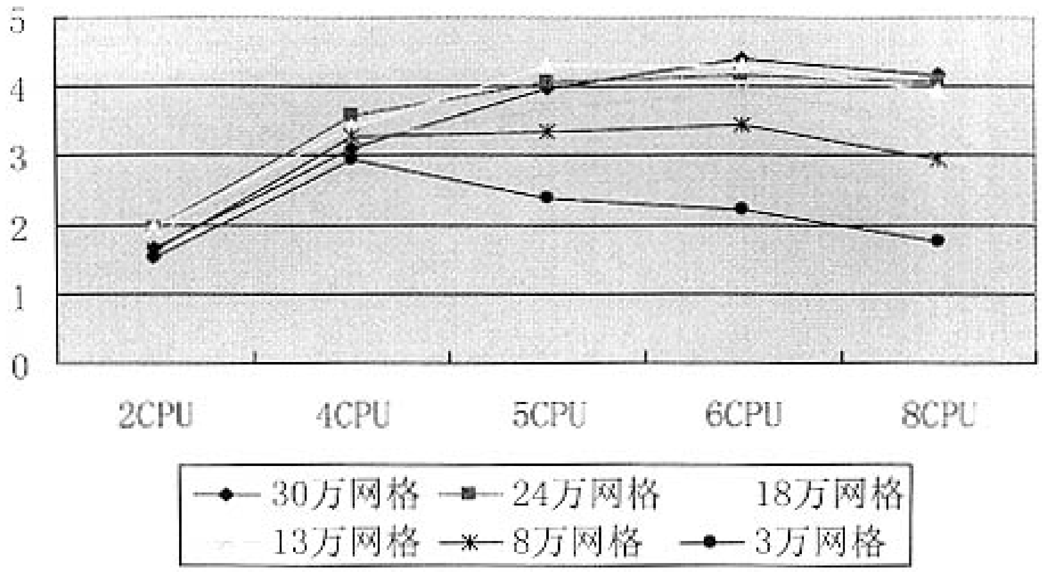


图 4- 11　Strip 方法的加速比分析图

4.3.2.3　效率分析

表 4- 16　Strip 方法的效率

结点数 / 网格数	2	4	5	6	8
30万网格	83.5	77.75	79.2	73.5	52
24万网格	97.5	89.5	81.4	69.5	50.88
18万网格	97	85.25	85.8	71.83	49.38
13万网格	92.5	81.25	73.2	65.5	43.5
8万网格	81	82	66.8	57.5	36.75
3万网格	76	74	48	36.83	22.13

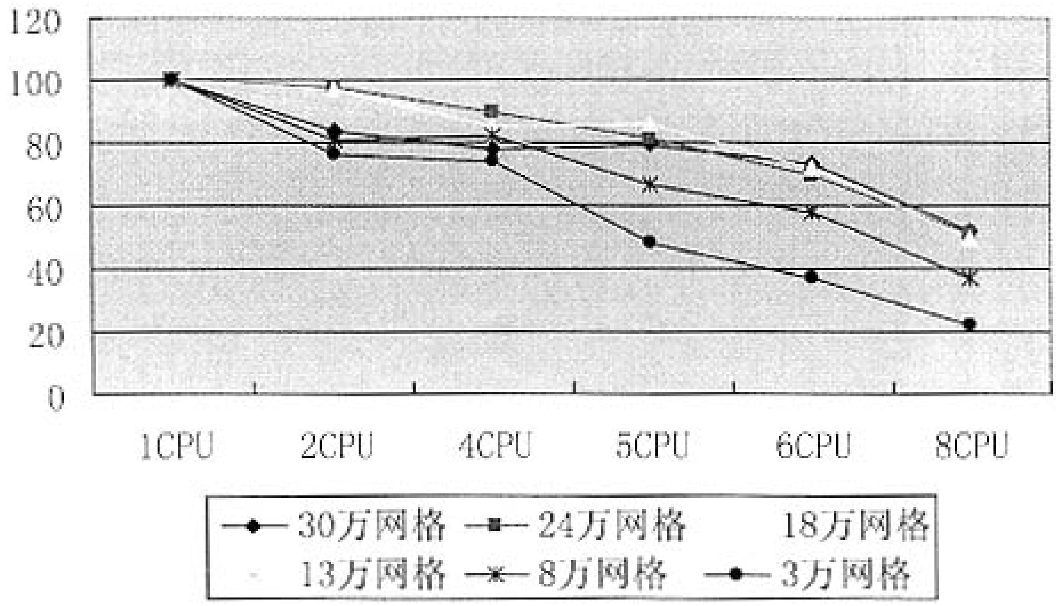


图 4- 12　Strip 方法的效率分析图

表 4-16 为 Principle Strip 方法的效率表，图 4-12 是其图形显示。

4.3.3 两种方法的比较

使用 3 万到 30 万的网格数，比较两种分区方法得到的结果类似，曲线的整体走势是一致的，没有很大的区别，经过分析得出，是因为网格数太少，致使两种方法的区别甚小。

4.4 小结

本章是数值模拟的最后一部分——后处理，对计算得到的数据进行图形可视化和分析，比较加速比和效率。

下图是初次计算得到的值，图中在两个处理器的时候，有些情况的效率值高于 100%，理论上是根本不可能的。经过分析，原因是在用单 CPU 计算时，工作站上另外还运行着其他的任务，导致串行执行时间比实际上的要长，使得加速比变高，而效率也就超过了 100%。所以，在确定工作站空闲情况下，重新计算了串行情况下的情况，得到了可行的结果，如上面文中所述。

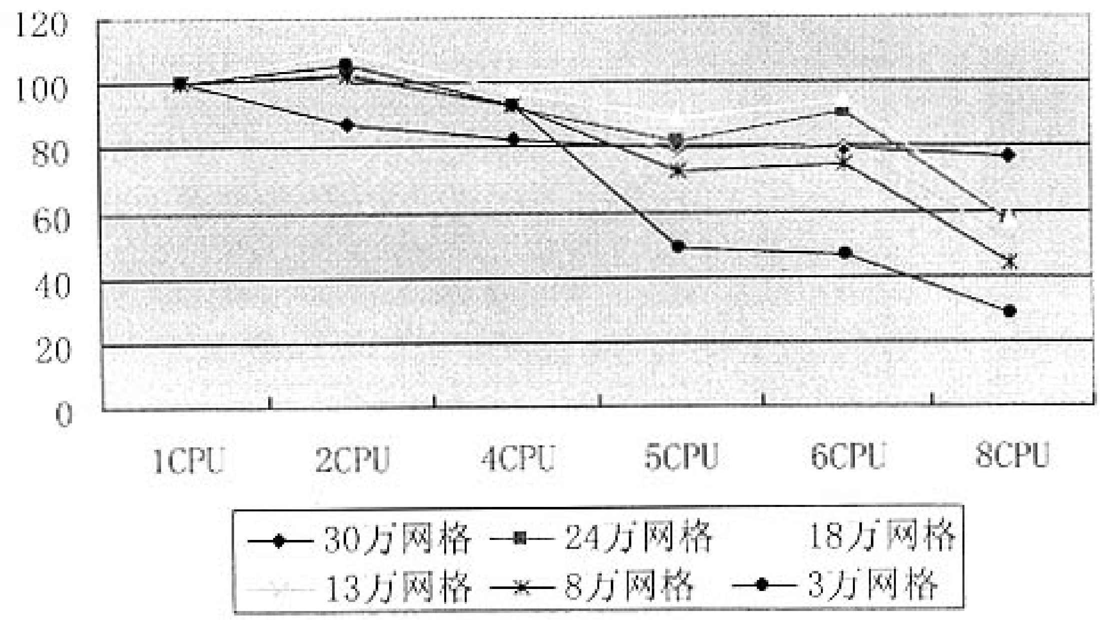


图 4- 13　有严重误差的效率示意图

第 5 章 总结

5.1 总结

通过对汽车外流场进行研究，用 Gambit 软件建立模拟计算的数学模型，确定边界条件，并且进行了网格剖分，用 Fluent 软件模拟出汽车外流场的流场分布情况，计算结果用 Fluent 软件和 Tecplot 软件进行处理，用图形来显示。计算的过程用课题组自己组建的 COW 并行环境对流场做了并行计算，得到大量的数据，并对结果进行加速比和效率的分析。为大规模的流体并行计算提供了理论依据。

计算得到的结果有以下几点：

1．流场方面：

汽车流场的流体线条很平滑，说明汽车外形设计比较合理。流场中，空气流在汽车顶部的速度比较大，在后窗和车尾两边的速度也比较大；而车尾中间速度比较小。压力则是车头和前窗玻璃的地方比较大。

2．并行计算方面：

(1) 执行时间：随着网格数的增加，每步迭代执行的时间也是增加的；网格数越多，随着处理器的增加，每步执行时间缩短的比例也越明显；随着处理器数的增加，每步迭代时间减小，但是并不是无限制的减小下去，当到了某个数目，之后反而会有所回升。

(2) 加速比：随着处理器的增加，加速比也会增大，但不是无限制的增大下去，当处理器继续增大到某个特定值时其加速比势必也会达到最大值，当处理器再继续增大，其加速比势必会降低。

(3) 效率：效率也就是处理器的利用率。对于固定规模的计算量，由于并行计算时有通信开销，处理器越多，通信开销越大，处理器的效率越低；对于计算规模不固定，则计算量越大，使用相同处理器时处理器的效率越高。

(4) 两种分区方法：由于计算规模不够大，所以由 Principle Axes 方法和 Principle Strip 方法分区得到的计算结果没有明显的区别。所以，这两种分区方法对于几十万网格数的情况没有太大区别。

5.2 进一步工作及展望

1．汽车模型做得比较简单，与实际的问题相差很多，进一步可以将模型复杂化，比如，加上车轮、后视镜等等，这样算出来的流场更加准确。

2．汽车的外流场分为层流区和湍流区。层流区主要分布在车前方无穷远处

至车顶部，不存在气流的分离流动；而湍流区主要分布在车顶部后缘至车后无穷远处，此处由于车身截面的突变造成了气流分离，分离流与底部及顶部两侧气流汇合，产生了复杂的尾部涡系。描述此处流动需引入湍流经验方程与带脉动时均量的 N-S 方程耦合，而此联立的方程组比较复杂，可以做进一步的研究。

3．虽然商用软件 Fluent 做的非常好，但是有时候用户想根据自己的情况，添加方程或者定义外界条件等。Fluent 有用户自定义函数(Udf—User Defined Function)这个功能，使用户可以自己编程定义方程以及材料属性等，然后导入 Fluent 进行编译，仍然可以使用 Fluent 强大的计算以及后处理功能。甚至，如果有新的好的并行算法，也可以用 Udf 将新的并行算法添加到 Fluent 中。将来可以做二次开发。

4．本实验的后处理得到的图是静态图，以后可以用 Tecplot 实现动画的效果，那么会使得结果更加清晰易懂。

5．本文的网格数目最大为 30 万，对于本文介绍的两种分区方法，差别不明显，所以可以进一步增加网格的数目，再对两种分区方法加以比较。并行计算方面进一步可以评测其可扩放性。

参考文献

[1] 马铁犹.计算流体动力学[M].北京:北京航空学院出版社,1986,6.

[2] L.M.米尔恩-汤姆森[美], 李裕文等译. 理论流体力学[M]. 北京. 机械工业出版社. 1984.

[3] 傅德薰. 流体力学数值模拟. 北京. 国防工业出版社. 1993.

[4] 熊鳌愧，詹德新.不可压流场数值模拟方法综述[J].武汉交通科技大学学报，1999.10.

[5] 宋振寰， 朱炳哲.计算流体力学在模拟汽车外流场研究方面的发展状况[J].汽车科技,2002.1.

[6] 吕明忠,高敦岳,傅立敏.计算机数值仿真在汽车外流场分析方面的应用研究[J].计算机辅助设计与图形学学报,2002.2.

[7] 吴军，钟志华，谷正气.汽车外流场数值仿真的进一步研究[J].机械工程学报,2003.9.

[8] 傅立敏,沈俊,王靖宇. 汽车流场及尾部涡系数值模拟[J]. 吉林工业大学自然科学学报,2000.4.

[9] 欧阳鸿武,杨晓辉.汽车外流场数值模拟中网格生产方法的改进[J].长沙交通学院学报[J],1999.9.

[10] 朱国林,徐庆新.计算流体力学并行计算技术研究综述[J].空气动力学学报,2003.2.

[11] Dirk Roose， Rafacl Van Driessche.并行计算机和计算流体力学并行算法[J].力学进展， 1998.2.25.

[12] 袁俊. 并行计算流体力学的研究与应用[J]. 上海超级计算中心,http://www.ssc.net.cn/xslw/bx.htm,2002.

[13] 都志辉编著. 高性能计算并行编程技术－MPI 并行程序设计[M]. 清华大学出版社, 2001.

[14] 黄铠, 徐志伟著. 可扩展并行计算－技术、结构与编程[M]. 机械工业出版社. 2000.

[15] 张丽君, 陈增荣, 高坤敏. 并行算法设计与分析[M]. 湖南科技出版社, 1984.

[16] 陈国良.并行计算－结构、算法、编程[M].北京：高等教育出版社， 1999.

[17] Cebeci, T. and Smith, A.M.O. Analysis of Turbulent Boundary Layers. Academic Press. London. 1974.

[18] Saffman, P.G. and Wilcox, D.C. Turbulent-Model Prediction for Turbulent Boundary Layers. AIAAJ. 1974. No.4, Vol.12: 541~546.

[19] B.E.Launder, D.B.Spalding, The Numerical Computation of Turbulent Flows. Comp. Methods Appl. Mech. Eng. 1974,Vol.3:269-289.

[20] K. A. Hoffman, S. T. Chiang. Computational Fluid Dynamics for Engineers. A Publication of Engineering Education System. Wichita, Kansas, USA. July, 1993.

[21] S.V. Patankar. Numerical Heat Transfer and Fluid Flow. New York. Hemisphere Pub. Corp. 1980.

[22] B.E.Launder, D.B.Spalding, Lectures in Mathematical Models of Turbulence. Academic Press, London. 1972:78-115.

[23] Li You Zhang. Turbulent Reacting Flow Equations of ψ—ω Method with Geral Orthogonal Curvilinear Coordinates System. Transport. Phenomena Science & Technology. 1992:797.

[24] Reynold, W.C. Computation of Turbulent Flows. Ann. Rev. Fluid Mech .1976, vol.8: 183.

[25] 李树民,朱国林,王开春. 基于分区的隐式求解二维不可压 NS 方程的并行实现[J].空气动力学学报,2002,3(增刊) :03906.

[26] 李勇，刘志友，安亦然.介绍计算流体力学通用软件——Fluent[M].水动力学研究与进展,2001.6.

[27] 张宝琳, 谷同祥, 数值并行计算原理与方法[M]. 国防工业出版社, 1999.

[28] 陈景良, 并行数值方法[M], 北京:清华大学出版社, 1983.

[29] 孙安香,宋君强,伍湘君. 并行计算的数据分配[J].计算机工程与科学,1997,5.

[30] 余舜京,徐田荣,邬维日. CFD 大型软件系统分布式并行处理模型研究[J].空气动力学学报,2002,3(增刊):11205.

[31] 何有世,袁寿其,王大承,丛小青. 计算流体力学(CFD)中的迭代法及其并行计算方法[J].中国安全科学学报,2002,6.

[32] 王正华,赵文涛. 二维不可压流程序在 PVM 环境下的并行计算[J].空气动力学学报,2002,3(增刊) :05905.

[33] Shahrouz Aliabadi, Khalil Shujaee, Tayfun Tezduyar. Parallel Simulation of Two-phase Flow Problems Using the Finite Element Method[R]. the 7th Symposium on the Frontiers of Massively Parallel Computation, Annapolis,Maryland,1999: p. 113.

[34] A.M.Baker, S.X.Ying. A Fixed Time Performance Evaluation of Parallel CFD Applications[J].IEEE,1994:18-23.

[35] 李国杰. 高速科学计算与大规模并行机. 计算物理. 1992, No.4,vol.9: 346～350.

[36] 王家谟、沈毅主编. 并行计算方法[M]. 国防工业出版社, 1987.

[37] 李晓梅等编著. 并行计算与偏微分方程数值解[M]. 国防科技大学出版社, 1990.

[38] 陈景仁［美］. 湍流模型及有限分析法[M]. 上海交通大学出版社. 1989:262~266.

作者攻读学位论文期间公开发表的论文

1. 辛晓华，张武，周华. 基于Fluent的绕流问题的数值模拟与并行计算，计算机工程与设计，已发表。

致　谢

本文是在我的导师张武博士的指导下完成的。在二年半的时间当中，我自始至终都得到了张老师的关怀和悉心的指导。无论是学习、研究还是生活上，他都给了我极大的启迪和帮助。同时张老师渊博的知识、深厚的数学功底、严谨的治学态度、执着的探索精神使我受益非浅，并激励我不断地学习和追求。在他的关心、信任和培养下，我已经从一个入校时知之甚少、依赖性较强的大学生，演变成了一名能独立完成一项科研任务，并有一定实践经验的科研工作者。同时也使我顺利地完成了二年半的学业。在这里我要对张老师表示我最深的谢意。

感谢上海大学力学所周华博士后以及北京大学力学系程暮林同学，在本课题的流体力学方面给予了极大的帮助。特别感谢程暮林同学的鼓励和耐心的帮助！

感谢高性能研究室的全体老师，在科研和生活中给予我关怀与支持。

在我的科研工作中，我还与同实验室其他研究生进行了良好的合作。另外，我还得到了他们的帮助，特别是杨林峰和郭晓燕等同学，在我的论文排版等方面都作了许多的工作并提了许多建议，在此一起表示感谢。

感谢网络中认识的网友以及论坛里的各位高手，他们在我的科研工作中很多细致的工作给予了很大的帮助。

感谢上海大学计算机工程与科学学院，我的科研成果离不开她的精心教育与培养。

感谢一直以来默默地关心、支持我的家人和朋友，是他们给予了我不断学习、进取的目标和动力。本文的研究工作成果，和他们的关怀帮助是分不开的。

再次向本论文工作所涉及的在这里提到和未提到的师长、同学表示诚挚的感谢。